Fully Auditable Privacy-preserving Cryptocurrency Against Malicious Auditors

Wulu Li[1], Yongcan Wang[1], Lei Chen[1], Xin Lai[1], Xiao Zhang[1], and Jiajun Xin[1]

Onething Technologies Co., Ltd., Shenzhen, China
liwulu@onething.net

Abstract. Privacy protection techniques have been thoroughly studied in the current blockchain research field with the famous representatives such as Monero and Zerocash, which have realized fully anonymous and confidential transactions. However, lack of audit can lead to abuse of privacy, and can help bad guys to conduct illegal activities, such as money laundering, transfer of illegal assets, illegal transactions, etc. Therefore, it is crucial to study the privacy-preserving cryptocurrency with full auditability. In this paper, under the framework similar to Monero, we propose FAPC, a fully auditable privacy-preserving cryptocurrency with security against malicious auditors. FAPC mainly consists of three schemes: a traceable and linkable ring signature scheme (TLRS), a traceable range proof (TRP), and a tracing scheme for long-term address (TSLA). In FAPC, the identities of UTXOs, transaction amounts and the corresponding long-term addresses can be traced by the auditor with maintaining anonymous and confidential to others. The constructions of TLRS and TRP are simple and modular, which only use standard ring signature as component, without any additional one-time signatures or zero-knowledge proofs. The TSLA is constructed by usage of standard ring signature and ElGamal encryption to realize traceability of long-term addresses in transactions. Moreover, all the schemes are secure against malicious auditors to realize a closer approach towards decentralization. We also give the security proofs and implementations of our schemes, as well as the performance results.

Keywords: Auditable blockchain · Privacy protection · Decentralization · Traceable and linkable ring signature · Traceable range proof · Tracing scheme for long-term address · Malicious auditor.

1 Introduction

Blockchain technology was first proposed by Nakamoto[25] in 2008. It is an application system that combines multiple underlying techniques including P2P networks, distributed data storage, network consensus protocols and cryptographic algorithms. It has features of open, transparentness, non-tamperability, traceability, and has various applications such as cryptocurrency (including Bitcoin[25], Ethereum [9], Monero[32, 26], Zerocash[29], etc.), anti-counterfeiting,

credit deposit and medical health, etc. In June 2019, Facebook announced "Libra"[14], an international blockchain-based cryptocurrency to support efficient cross-border transactions with openness and equality.

In blockchain theory, privacy-preserving techniques has been developed in this decade to provide a potential replacement of traditional blockchain-based cryptocurrencies such as Bitcoin[25] and Ethereum[9], to support various privacy-preserving scenarios such as salary, donation, bidding, taxation, etc. A series of works have been proposed during these years such as Confidential Transaction[23], Dash[13], Mimblewimble[19], Monero[32, 26] and Zerocash[29], etc. Among all the privacy-preserving cryptocurrencies, Monero has realized fully anonymous and confidential transactions, which can protect the privacy of identities for both initiators and recipients in transactions, as well as the transaction amount. In contrast, Zerocash is deeply related with the zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs), which provides the preimage proof of hash commitment, and therefore achieves fully privacy of identity and amount. Nevertheless, zk-SNARKs technique uses *common reference string* (CRS) with GB size and it is based on non-falsifiable assumptions, that weakens its potential competitiveness compared to other candidates.

However, existing privacy-preserving cryptocurrencies have no regulatory functions so as to cause potential risks of illegal activities such as money laundering, transfer of illegal assets, illegal transactions, frauds, etc. Bad guys can easily escape from audit to conduct illegal activities without any punishments. Meanwhile, in the application of privacy-preserving cryptocurrency, auditors and policy-making institutions need to have a comprehensive understanding of the economic operation and development, they need to recover the addresses and amounts in transactions. Therefore, it is crucial to develop privacy-preserving cryptocurrency with regulatory functions to support both privacy protection and auditability for money flows, users' addresses and amounts in all transactions. Moreover, the auditable privacy-preserving blockchain system needs to keep secure when the auditor is corrupted or malicious, which means the auditor can only trace the money flows, addresses and amounts in transactions, but cannot double spend, corrupt users, slander honest users or escape from audit.

1.1 Our Contributions

In this paper, we propose FAPC: the first (to the best of our knowledge) construction of **F**ully **A**uditable **P**rivacy-preserving **C**ryptocurrency against malicious auditors to achieve both privacy protection and auditability for money flows, users' addresses and transaction amounts. Under the framework similar to Monero, FAPC mainly consists three schemes: a **T**raceable and **L**inkable **R**ing **S**ignature (TLRS), a **T**raceable **R**ange **P**roof (TRP) and a **T**racing **S**cheme for **L**ong-term **A**ddress (TSLA).

In the construction of FAPC, under UTXO model (same as Monero), linkable ring signature is replaced by TLRS, range proof is replaced by TRP and the key generation algorithm of every UTXO for the recipient is replaced by TSLA. The construction of FAPC is modular, we use ring signature as the key component in

TLRS, TRP and TSLA, we can choose the most suited ring signature schemes (most efficient or most compact) for different applications and requirements.

In FAPC, there exists an auditor in the system with possession of the audit trapdoor to trace the identities of UTXOs (money flows) in ring signatures, the addresses of initiators and recipients, and the amounts in all transactions. Moreover, in FAPC, the works for verification and audit are independent, which means that the auditor is not responsible for the verification of transactions. The system can run safely and correctly when the auditors are not involved, anyone (including the malicious auditor) cannot generate a valid transaction to make the auditor fail to trace. The design idea of FAPC is a closer approach to meet the characteristic of "decentralization". The power of auditor in FAPC is strictly restricted, who can only trace the private information in the transactions, while cannot conduct any illegal activities to break the security.

Traceable and Linkable Ring Signature The traceable and linkable ring signature scheme is directly from the linkable ring signature scheme with an additional security requirement called traceability, to ensure the identity of signer can be traced by the auditor, while the linkability remains the same that two ring signatures can be linked if they are signed by the same signer. We give a simple and modular construction of TLRS which only uses ring signature as component, without any additional one-time signatures or zero-knowledge proofs. Actually, in the construction of TLRS, ring signature can be directly switched to traceable and linkable ring signature by a randomized combination between public key set and tracing keys, which will be another independent point of interest. We give a brief description of TLRS in the following:

1. The public parameters are $(\mathbb{G}, q, g, h_1 = g^y, h_2)$, where g (uniformly generated by system) is a generator of an elliptic curve $\mathbb{G}$ with prime order q, y is the audit trapdoor, generated by the auditor, h_2 is another generator of $\mathbb{G}$ with its discrete logarithm to g being unknown to anyone.
2. Every user generates his $(\mathsf{PK}, \mathsf{SK})$ by usage of public parameters, the key generation algorithm remains the same as the linkable ring signature in Monero.
3. When signing, the signer publishes a tracing key TK and a key-image I, computes a new public keys set L_{RPK} for ring signature, then generates the ring signature τ by usage of his private key SK, the basis element (generator) for ring signature is not fixed, which is different from Monero.
4. The verifier checks whether I is already in key-image set to determine whether double signing (double spending) occurs. Then computes the ring signature public keys set L_{RPK} and checks the validity of the ring signature τ, then outputs the verification result.
5. The auditor can trace the identity of signer by usage of the trapdoor y and the tracing key TK.

In TLRS, the key-image is used in the linkability check to examine whether double signing occurs, the tracing key is used in the audit algorithm to trace

the identity of signer. TLRS has anonymity, unforgeability, linkability, nonslanderability and traceability, under the hardness assumption of discrete logarithm, DDH assumption and the security of ring signature. Moreover, for a malicious auditor (or any adversary with possession of trapdoor), he can only break the anonymity of TLRS, but cannot generate illegal TLRS signatures (double spend), cannot slander other users (make other valid TLRS signatures illegal) or break the traceability (escape from audit). This is a closer approach to meet the characteristic of "decentralization".

Traceable Range Proof Traceable range proof is a special variant of range proof, in which there is an auditor who can use the trapdoor to trace the transaction amount. The zero-knowledge property of traceable range proof only holds for adversary without possession of trapdoor. We give the first (to the best of our knowledge) construction of traceable range proof by usage of Borromean range proof[26] and Pedersen commitment[27]:

In the construction of TRP, we use Pedersen commitment $c = g^x h_2^a$ to hide the transaction amount a, the auditor generates his trapdoor y and computes $h_1 = g^y$, the public parameters are $(\mathbb{G}, q, g, h_1 = g^y, h_2)$, h_2 is another generator of $\mathbb{G}$ with its discrete logarithm to g is unknown to anyone, similar to TLRS. For a's binary expansion $a = a_0 + 2a_1 + \cdots + 2^{n-1}a_{n-1}$, for every bit $i = 0, \cdots, n-1$, prover computes a tracing key TK_i and a tag I_i, prover also generates a ring L^i_{PK} with two elements and generates the Borromean multi-ring signature for n rings. The verifier only need to check the validity of Borromean ring signature and the correctness of the binary expansion. The auditor can trace $a_i = 0$ *or* 1 for every $i = 0, \cdots, n-1$ to recover the total amount $a = a_0 + 2a_1 + \cdots + 2^{n-1}a_{n-1}$ by usage of the trapdoor y and $\{\mathsf{TK}_i\}_{i=0,\cdots,n-1}$.

In our construction of TRP, the auditor cannot compute x form $c = g^x h_2^a$, which partially protects the privacy of users, and it is a balance between audit and privacy protection. TRP has completeness, soundness, zero-knowledge and traceability against malicious auditors (except for the zero-knowledge property), under the hardness assumption of discrete logarithm, DDH assumption and the security of Borromean ring signature.

Tracing Scheme for Long-term Address In FAPC, every user's address consists of a view public key $A = g^{x_v}$ and a spending public key $B = g^{x_s}$, same as Monero in Cryptonote[32]. We construct TSLA to achieve the address anonymity and traceability in FAPC transactions. The public parameters are $(\mathbb{G}, q, g, h_1 = g^y, h_2)$, similar to TLRS. When paying, the initiator chooses another $l-1$ users' addresses, together with the real recipient's address, to generate an address list $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$, then he encrypts the secret information, computes the public key PK_{out} of the new UTXO, then generates a position-preserving double-ring signature to prove the validity of PK_{out}, where position-preserving means that the position of secret key in each ring is same. The recipient can find his transaction and recover the secret key SK_{out} of the new UTXO by usage of his address secret key ASK. The auditor can trace the

recipient's address by decryption with the trapdoor. For a malicious auditor with possession of the trapdoor, he cannot make a valid transaction to escape from audit, while the address of recipient remains anonymous to others. Since the auditor can trace the recipient addresses in all transactions, then he can also recover the corresponding initiator addresses immediately with the help of money flow traceability by TLRS.

1.2 Related Works

Ring Signatures Ring signature is a special type of signature scheme, in which signer can sign on behalf of a group chosen by himself, while maintaining anonymous within the group. In ring signatures, signer selects a list of public keys L_{PK}=$\{PK_1, \cdots, PK_m\}$ as the ring elements, and uses his secret key SK_κ to sign, where $\kappa \in \{1, \cdots, m\}$. Verifier cannot determine the signer's identity. Ring signature was first proposed by Rivest, Shamir and Tauman[28] in 2001, they constructed ring signature schemes based on RSA trapdoor permutation and Robin trapdoor function, in the random oracle model. In 2002, Abe *et al.*[1] proposed AOS ring signature, which simultaneously supported discrete logarithm (via Sigma protocol) and RSA trapdoor functions (via hash and sign), also in the random oracle model. In 2006, Bender *et al.*[5] introduced the first ring signature scheme in the standard model, by making use of pairing technique. In 2015, Maxwell *et al.*[24] gave Borromean signature scheme, which is a multi-ring signature based on AOS with signature size reduced from $mn + n$ to $mn + 1$, where m denotes the ring size and n denotes the number of rings. It's worth emphasizing that the signature sizes in these schemes are linear to the number of ring elements.

In 2004, building from RSA accumulator, Dodis *et al.*[12] proposed a ring signature scheme with constant signature size in the random oracle model. In 2007, Chandran *et al.*[10] gave a standard model ring signature scheme with $O(\sqrt{m})$ signature size, using pairing technique and CRS. In 2015, under the discrete logarithm assumption, Groth *et al.*[18] introduced a ring signature scheme with $O(\log m)$ signature size, in the random oracle model.

Linkable Ring Signatures Linkable ring signature is a variant of ring signature, in which the identity of the signer in a ring signature remains anonymous, but two ring signatures can be linked if they are signed by the same signer. Linkable ring signatures are suitable in many different practical applications such as privacy-preserving cryptocurrency (Monero), e-Voting, cloud data storage security, etc. In Monero, linkability is used to check whether double spending happens. The first linkable ring signature scheme is proposed by Liu *et al.*[22] in 2004, under discrete logarithm assumption, in the random oracle model. Later, Tsang *et al.*[31] and Au *et al.*[2] proposed accumulator-based linkable ring signatures with constant signature size. In 2013, Yuen *et al.*[35] gave a standard model linkable ring signature scheme with $O(\sqrt{n})$ signature size, from pairing technique. In 2014, Liu *et al.*[21] gave a linkable ring signature with unconditional

anonymity, he also gave the formalized security model of linkable ring signature, which we will follow in this paper. In 2015, Back *et al.*[3] proposed an efficient linkable ring signature scheme LSAG, which shortens the signature size of [22]. In 2016, based on work of Fujisaki *et al.*[16], Noether *et al.*[26] gave a linkable multi-ring signature scheme MLSAG, which supports transactions with multiple inputs, and was used by Monero. In 2017, Sun *et al.*[30] proposed Ring-CT 2.0, which is an accumulator-based linkable ring signature with asymptotic smaller signature size than Ring CT, but is less competitive when n is small. In 2019, Yuen *et al.*[36] proposed Ring-CT 3.0, a modified Bulletproof-based 1-out-of-n proof protocol with logarithmic size, which has the functionality of (linkable) ring signature. In 2019, Goodell *et al.*[17] proposed CLSAG, which improved the efficiency of MLSAG.

Range Proofs Range proof is a special zero-knowledge proof to prove a committed hidden amount a lies within a certain range $[0, 2^n - 1]$ without revealing the amount. The Pedersen-commitment-based range proofs are used in Monero system. In 2016, Neother *et al.*[26] gave the Borromean range proof, building from the Borromean ring signature[24], with linear proof size to the binary length of range. In 2018, Bünz *et al.*[7] introduced Bulletproofs, an efficient non-interactive zero-knowledge proof protocol with short proofs and without a trusted setup, the proof size is only logarithmic to the witness size and it is used in projects such as Monero and DERO[11]. There are also privacy-preserving blockchain systems such as Qiusqius[15], Zether[6] with different commitments and range proofs. Moreover, range proof can also be built from zero-knowledge for arithmetic circuits, including [33, 34, 4, 8].

Traceable Privacy-preserving Cryptocurrencies In 2019, Li *et al.*[20] gives a construction of traceable Monero to achieve anonymity and traceability of signers' identities and users' addresses, by usage of paring-based accumulator, signature of knowledge, zk-SNARKs and verifiable encryption from Ring-CT 2.0. Their construction provides the same functionality as TLRS and TSLA, but relies on a trusted setup and CRS, which is inefficient for computation and storage.

In this paper, we give the constructions of FAPC, including TLRS, TRP and TSLA with unconditional traceability of signers' identities, transaction amounts and long-term addresses, with better efficiency and standard assumptions. Compared to [20], FAPC has four main advantages:

1. The construction of FAPC is modular, the elliptic-curve-based ring signature scheme is used as the key component in TLRS, TRP and TSLA. The choice of ring signature is not restricted and we can choose the most suited ones in different applications;
2. FAPC has the functionality of full auditability, which means that the auditor can not only trace the identities of UTXOs (money flows) and long-term addresses, but also trace the transaction amounts, which are untraceable in [20];

3. The security of FAPC (including TLRS, TRP and TSLA) relies on standard assumptions (paring-free) and the security of ring signatures, without any trusted setup or CRS;
4. FAPC is very efficient in transaction generation, verification and audit, the computation time for a whole transaction (1 input UTXO, 2 output UTXO) is less than 40ms. While in [20], it is required over 1s for a whole transaction (1 input UTXO, 2 output UTXO).

1.3 Paper Organization

In section 2 we give some preliminaries; in section 3 we give the construction of FAPC, and introduce the security requirements; in section 4 we give the construction and security proof of the traceable and linkable ring signature (TLRS); in section 5 we give the constructions and security proof of traceable range proof (TRP); in section 6 we introduce the tracing scheme for long-term address (T-SLA) and the security proofs; in section 7 we introduce the implementation and performance of FAPC; in section 8 we give the conclusion.

2 Preliminaries

2.1 Notations

In this paper, we use multiplicative cyclic group $\mathbb{G}$ to represent elliptic group with prime order $|\mathbb{G}| = q$, g is the generator of $\mathbb{G}$, group multiplication is $g_1 \cdot g_2 = g_1 g_2$ and exponentiation is g^a. $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ is the set of nonzero elements in $\mathbb{Z}_q$. We use $H(\cdot)$ to represent hash function, use $H_p(\cdot)$ to represent Hash-to-Point, and $negl(\cdot)$ to represent negligible functions. For verifiers, 1 is for *accept* and 0 is for *reject*. For adversaries, PPT means probabilistic polynomial time. The DDH assumption means any PPT adversary cannot distinguish (g^a, h^a) from (g^a, h^r), where r is uniformly sampled from $\mathbb{Z}_q^*$. The hardness of discrete logarithm problem means that any PPT adversary cannot compute x from g^x. Oracle $\mathcal{RO}$ refers to the random oracle. The security parameter of this paper is $\lambda = \lceil \log q \rceil$, where $q = |\mathbb{G}|$.

2.2 Ring Signatures

Ring signature scheme usually consists of four algorithms: Setup, KeyGen, Rsign, and Verify.

- $\mathsf{Par} \leftarrow \mathsf{Setup}(\lambda)$ is a probabilistic polynomial time (PPT) algorithm which, on input a security parameter λ, outputs the set of security parameters Par which includes λ.
- $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(\mathsf{Par})$ is a PPT algorithm which, on input the security parameters Par, outputs a public/private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$.
- $\sigma \leftarrow \mathsf{Rsign}(\mathsf{SK}_\kappa, \mu, L_{\mathsf{PK}})$ is a ring signature algorithm which, on input user's secret key SK_κ, a list of users' public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$, where $\mathsf{PK}_\kappa \in L_{\mathsf{PK}}$, and a message μ, outputs a ring signature σ.

– $1/0 \leftarrow \mathsf{Verify}(\mu, \sigma, L_{\mathsf{PK}})$ is a verification algorithm which, on input a message μ, a list of users' public keys L_{PK} and a ring signature σ, outputs 1 or 0.

The security definition of ring signature contains *unforgeability* and *anonymity*. Before giving their definitions, we consider the following oracles which together model the ability of the adversaries in breaking the security of the schemes, in fact, the adversaries are allowed to query the four oracles below:

– $c \leftarrow \mathcal{RO}(a)$. *Random oracle*, on input a, random oracle returns a random value.
– $\mathsf{PK}_i \leftarrow \mathcal{JO}(\perp)$. *Joining oracle*, on request, adds a new user to the system. It returns the public key PK_i of the new user.
– $\mathsf{SK}_i \leftarrow \mathcal{CO}(\mathsf{PK}_i)$. *Corruption oracle*, on input a public key PK_i that is a query output of $\mathcal{JO}$, returns the corresponding private key SK_i.
– $\sigma \leftarrow \mathcal{SO}(\mathsf{PK}_\kappa, \mu, L_{\mathsf{PK}})$. *Signing oracle*, on input a list of users' public keys L_{PK}, the public key of the signer PK_κ, and a message μ, returns a valid ring signature σ.

Definition 1 (Unforgeability) *Unforgeability for ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ wins the game if he successfully forges a ring signature $(\sigma^*, L^*_{PK}, \mu^*)$ satisfying the following:*

1. $\mathsf{Verify}(\sigma^*, L^*_{PK}, \mu^*) = 1$.
2. *Every $\mathsf{PK}_i \in L^*_{PK}$ is returned by $\mathcal{A}$ to $\mathcal{JO}$.*
3. *No $\mathsf{PK}_i \in L^*_{PK}$ is queried by $\mathcal{A}$ to $\mathcal{CO}$.*
4. *(μ^*, L^*_{PK}) is not queried by $\mathcal{A}$ to $\mathcal{SO}$.*

We give the advantage of $\mathcal{A}$ in forging attack as follows:

$$\mathrm{Adv}^{forge}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins}].$$

A ring signature scheme is unforgeable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{forge}_{\mathcal{A}} = negl(\lambda)$.

Definition 2 (Anonymity) *Anonymity for ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$ and $\mathcal{CO}$. $\mathcal{A}$ gives a set of public keys $L_{PK} = \{PK_1, \cdots, PK_m\}$, $\mathcal{S}$ randomly picks $\kappa \in \{1, \cdots, m\}$, computes $\sigma = \mathsf{Rsign}(SK_\kappa, \mu, L_{PK})$ and sends σ to $\mathcal{A}$, where SK_κ is the corresponding private key of PK_κ, then $\mathcal{A}$ outputs a guess $\kappa^* \in \{1, \cdots, m\}$. $\mathcal{A}$ wins the game if he successfully guesses $\kappa^* = \kappa$.*

We give the advantage of $\mathcal{A}$ in anonymity attack as follows:

$$\mathrm{Adv}^{anon}_{\mathcal{A}} = |\Pr[\kappa^* = \kappa] - 1/m|.$$

A ring signature scheme is anonymous if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{anon}_{\mathcal{A}} = negl(\lambda)$.

In the construction of TLRS, we use ring signature (unforgeable and anonymous in the random oracle model, simulatable by programming the random oracle) as component, we may select AOS scheme[1] (linear size) or Ring-CT 3.0[36] (logarithmic size) in our construction. The choice of ring signature component is not restricted, we can choose the most suited ones (most efficient or most compact ones) for different ring sizes in different applications, the detailed descriptions of AOS and AOS' are in the Appendix A.

2.3 Linkable Ring Signatures

Compared to ring signature, linkable ring signature has the function of linkability, that is, when two ring signatures are signed by the same signer, they are linked by the algorithm Link. We give the definition of Link below:

– $linked/unlinked \leftarrow \mathsf{Link}((\sigma, \mu, L_{\mathsf{PK}}), (\sigma', \mu', L'_{\mathsf{PK}}))$: verifier checks the two ring signatures are linked or not, output the result.

The security definition of linkable ring signature contains *unforgeability*, *anonymity*, *linkability* and *nonslanderability*. The *unforgeability* is the same as Definition 1, and the *anonymity* is slightly different from Definition 2 with additional requirements that all public keys in L_{PK} are returned by $\mathcal{A}$ to $\mathcal{JO}$ and all public keys in L_{PK} are not queried by $\mathcal{A}$ to $\mathcal{CO}$ (if the adversary corrupts some of the public keys, then he can break the anonymity of the scheme by compute the corresponding key-images in advance). In the rest of this paper, we use this modified definition of *anonymity* in sTLRS and its security proof.

We give the definition of *linkability* and *nonslanderability* as follows:

Definition 3 (Linkability) *Linkability for linkable ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ wins the game if he successfully forges k ring signatures $(\sigma_i, L^i_{\mathsf{PK}}, \mu_i), i = 1, \cdots, k$, satisfying the following:*

1. *All σ_is are not returned by $\mathcal{A}$ to $\mathcal{SO}$.*
2. *All L^i_{PK} are returned by $\mathcal{A}$ to $\mathcal{JO}$.*
3. *$\mathsf{Verify}(\sigma_i, L^i_{\mathsf{PK}}, \mu_i) = 1, i = 1, \cdots, k$.*
4. *$\mathcal{A}$ queried $\mathcal{CO}$ less than k times.*
5. *$\mathsf{Link}((\sigma_i, L^i_{\mathsf{PK}}, \mu_i), (\sigma_j, L^j_{\mathsf{PK}}, \mu_j)) = unlinked$ for $i, j \in \{1, \cdots, k\}$ and $i \neq j$.*

We give the advantage of $\mathcal{A}$ in link attack as follows:

$$\mathrm{Adv}^{link}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins}].$$

A linkable ring signature scheme is linkable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{link}_{\mathcal{A}} = negl(\lambda)$.

The *nonslanderability* of a linkable ring signature scheme is that $\mathcal{A}$ cannot slander other honest users by generating a signature linked with signatures from honest users. In the following we give the definition:

Definition 4 (Nonslanderability) *Nonslanderability for linkable ring signature schemes is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$. $\mathcal{A}$ gives a list of public keys L_{PK}, a message μ and a public key $\mathsf{PK}_\kappa \in L_{PK}$ to $\mathcal{S}$, $\mathcal{S}$ returns the corresponding signature $\sigma \leftarrow \mathsf{Rsign}(\mathsf{SK}_\kappa, L_{PK}, \mu)$ to $\mathcal{A}$. $\mathcal{A}$ wins the game if he successfully outputs a ring signature $(\sigma^*, L^*_{PK}, \mu^*)$, satisfying the following:*

1. $\mathsf{Verify}(\sigma^*, L^*_{PK}, \mu^*) = 1$.
2. PK_κ *is not queried by* $\mathcal{A}$ *to* $\mathcal{CO}$.
3. PK_κ *is not queried by* $\mathcal{A}$ *as input to* $\mathcal{SO}$.
4. $\mathsf{Link}((\sigma, L_{PK}, \mu), (\sigma^*, L^*_{PK}, \mu^*)) = linked$.

We give the advantage of $\mathcal{A}$ in slandering attack as follows:

$$\mathrm{Adv}^{slander}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ wins}].$$

A linkable ring signature scheme is nonslanderable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{slander}_{\mathcal{A}} = negl(\lambda)$.

According to [21], linkability and nonslanderability imply unforgeability:

Lemma 5 ([21]) *If a linkable ring signature scheme is linkable and nonslanderable, then it is unforgeable.*

2.4 Traceable and Linkable Ring Signatures

Similar to the security definitions of linkable ring signature, a PPT adversary $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$, the security of TLRS contains unforgeability, anonymity, linkability, nonslanderability and traceability. Considering the existence of auditor, who can trace the identities of signers, so the anonymity only holds for adversary who not possesses the trapdoor. Moreover, the unforgeability, linkability, nonslanderability remain the same as in linkable ring signature. For a malicious auditor (or any adversary who corrupts the auditor), he cannot forge signatures of other users or break the linkability and nonslanderability of TLRS, which means that the malicious auditor cannot spend money of other users, double spend or slander other honest users.

TLRS has an additional security requirement called traceability, which enables auditor with ability to trace signers' identities, for any PPT adversary $\mathcal{A}$ with possession of trapdoor, he cannot escape from audit. We give the formal definition of traceability in the following:

Definition 6 (Traceability) *Traceability for traceable and linkable ring signature schemes (TLRS) is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide the public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$. $\mathcal{A}$ generates a list of public keys $L_{PK} = \{PK_1, \cdots, PK_m\}$, $\mathcal{A}$ wins the game if he successfully generates a valid TLRS signature (σ, L_{PK}, μ) using $PK_\kappa \in L_{PK}$, satisfying the following:*

1. $\mathsf{Verify}(\sigma, L_{PK}, \mu) = 1$.
2. $\mathsf{PK}_i \neq \mathsf{PK}_j$ *for* $1 \leq i < j \leq m$.
3. $\mathsf{Trace}(\sigma, y) \neq \kappa$ *or* $\mathsf{Trace}(\sigma, y) = \perp$.

We give the advantage of $\mathcal{A}$ *in tracing attack as follows:*

$$\mathrm{Adv}_{\mathcal{A}}^{trace} = \Pr[\mathcal{A} \text{ wins}].$$

TLRS scheme is traceable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{trace} = negl(\lambda)$.

2.5 Zero-knowledge proofs

Zero-knowledge proof system is a proof system (P, V) in which a prover proves to the verifier that he has a certain knowledge but does not reveal the knowledge itself. The formal definition is that given language L and relation R, for $\forall x \in L$, there exists a witness w such that $(x, w) \in R$, to prove $x \in L$ without disclosing w. The transcript between prover and verifier is $\langle P(x, w), V(x)\rangle$, the proof is correct (or wrong) if $\langle P(x, w), V(x)\rangle = 1(or\ 0)$. The security notions of zero-proof system contains *completeness*, *soundness* and *zero-knowledge*:

Definition 7 (Completeness) (P, V) *has* ***completeness*** *for any non-uniform polynomial time adversary* $\mathcal{A}$,

$$\begin{aligned}\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda) : (x, w) \notin R \text{ or } \langle P(x, w), V(x)\rangle = 1] \\ = 1 - negl(\lambda).\end{aligned}$$

When the probability equals 1, *then* (P, V) *has perfect completeness.*

Definition 8 (Soundness) (P, V) *has* ***soundness*** *for any non-uniform polynomial time adversary* $\mathcal{A}$ *and* $x \notin L$,

$$\mathsf{Pr}[(x, s) \leftarrow \mathcal{A}(1^\lambda) : \langle P(x, w), V(x)\rangle = 1] = negl(\lambda).$$

In Σ *protocols with Fiat-Shamir transformation in the random oracle model, we use the notion of* ***special soundness****, that is, for a* 3*-round interactive proof protocol, if a non-uniform polynomial time adversary* $\mathcal{A}$ *can generate* 2 *valid proofs* $(x, c, e_1, s_1), (x, c, e_2, s_2)$, *then there exists an extraction algorithm* Ext *which can extract a witness* $(x, w) \in R$, *where* c *represents the commitment,* e_is *are challenges and* s_is *are responses.*

Definition 9 (Zero-knowledge) (P, V) *has* ***perfect*** *(or* ***computational****)* ***zero-knowledge****, for any non-uniform polynomial time (or PPT) adversary* $\mathcal{A}$,

$$\begin{aligned}\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle P(x, w), V(x, \rho)\rangle : \\ (x, w) \in R \text{ and } \mathcal{A}(tr) = 1] \\ = (or \approx_c)\mathsf{Pr}[(x, w) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow S(x, \rho) : \\ (x, w) \in R \text{ and } \mathcal{A}(tr) = 1].\end{aligned}$$

In Fiat-Shamir-based protocol, the randomness of ρ *is from the output of hash function, it is said to be* ***public coin*** *and the protocol is* ***honest-verifier zero-knowledge****.*

Pedersen Commitment Pedersen commitment[27] was proposed in 1991, for an elliptic curve $(\mathbb{G}, q = |\mathbb{G}|, g, h)$, where g is a generator of $\mathbb{G}$, h is a random element with discrete logarithm unknown to anyone.

Definition 10 (Pedersen commitment) *The Pedersen commitment for a is $c = g^x h^a$, where $x \in \mathbb{Z}_q^*$ is a blinding element. Under the hardness of discrete logarithm, Pedersen commitment has the following properties:*

- *(Hiding) Any (computational unbounded) adversary $\mathcal{A}$ cannot distinguish $c = g^x h^a$ from $c' = g^{x'} h^{a'}$.*
- *(Binding) Any PPT adversary $\mathcal{A}$ cannot generate another secret a' binding with $c = g^x h^a = g^{x'} h^{a'}$.*
- *(Homomorphic) Given $c_1 = g^x h^a, c_2 = g^y h^b$, then $c_1 \cdot c_2 = g^{x+y} h^{a+b}$ is a new commitment for $a + b$.*

2.6 Range proofs

Range proof is a special type of zero-knowledge proof with security requirements including completeness, soundness and zero-knowledge. In Monero system, range proofs are used to hide the transaction amount and prove the validity of it (lies in a certain range). Borromean range proof is used in Monero with perfect completeness, special soundness and honest verifier zero-knowledge. We give the introduction of Borromean range proof in the Appendix A.

2.7 Traceable Range proofs

In a traceable range proof, considering the auditor who can trace the amounts of transactions, zero-knowledge only holds for adversary not possesses the trapdoor, while the completeness and soundness remains the same as in range proof for any PPT adversary $\mathcal{A}$. For traceable range proof, we need another security concept called **traceability**. Since traceable range proof enables auditor with ability to trace the hidden amounts of transactions, for any PPT adversary $\mathcal{A}$ (including the malicious auditor), it is necessary that he cannot escape from audit (making valid transaction with amount untraceable). We give the formal definition of traceability in the following:

Definition 11 (Traceability of TRP) *Traceability for traceable range proof is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracle $\mathcal{RO}$. $\mathcal{A}$ generates a commitment c for a hidden value a and the range proof $\pi(c)$, $\mathcal{A}$ wins the game if:*

1. $\mathsf{Verify}(c, \pi(c)) = 1$.
2. $\mathsf{Trace}(\pi(c), trapdoors) \neq a$.

We give the advantage of $\mathcal{A}$ in traceability attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{trace} = \Pr[\mathcal{A} \text{ wins}].$$

A traceable range proof is traceable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{trace} = negl(\lambda)$.

2.8 Linkable Multi-ring Signature in Monero

In Monero system, every UTXO has its public-private key pair ($\mathsf{PK} = g^s, \mathsf{SK} = s$) and the corresponding value commitment $c = g^x h^a$ (or $c = h^x g^a$ in Bulletproofs), where c is Pedersen commitment[27], a is the hidden value and x is the blinding element. In a transaction, the initiator Alice chooses $m-1$ hiding UTXOs: $\{(\mathsf{PK}_i, c_i = g^{x_i} h^{a_i})\}_{i=1,\cdots,m-1}$, along with her input UTXO ($\mathsf{PK}_A = g^s, c_A = g^{x_A} h^{a_A}$), to generate a set of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_A, \mathsf{PK}_1, \cdots, \mathsf{PK}_{m-1}\}$ (randomized order), Alice also generates the output UTXO ($\mathsf{PK}_B, c_B = g^{x_B} h^{a_B}$) (the generation algorithm of PK_B is in the Appendix), where the input value equals the output value $a_A = a_B$. Then Alice computes another ring of commitments (same order as in L_{PK}):

$$L_v = \{c_A c_B^{-1}, c_1 c_B^{-1}, \cdots, c_{m-1} c_B^{-1}\}$$

$$= \{g^{x_A - x_B}, g^{x_1 - x_B} h^{a_1 - a_B}, \cdots, g^{x_{m-1} - x_B} h^{a_{m-1} - a_B}\}.$$

Alice uses linkable 2-ring signature to sign the transaction by L_{PK} and L_v, with the same position of signing key in each ring, we call it the position-preserving linkable multi-ring signature. In this paper, we use TLRS to construct an efficient and compact multi-ring signature in the application of FAPC.

3 Description of FAPC

In this section we give the description of FAPC, the first fully auditable privacy-preserving cryptocurrency against malicious auditors. FAPC achieves both privacy protection and auditability for money flows, users' addresses and transaction amounts. We give the construction in 3.1 and discuss the security requirements in 3.2.

3.1 Construction

The building blocks of FAPC consists of a traceable and linkable ring signature scheme (TLRS), a traceable range proof (TRP) and a tracing scheme for long-term address (TSLA), under the framework of Monero in the UTXO model.

Algorithm 1: FAPC

$\mathsf{Par} \leftarrow \mathsf{FAPC.Setup}(\lambda)$:

1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, the auditor generates $y \in \mathbb{Z}_q$ as the trapdoor, computes $h_1 = g^y$, system computes $h_2 = H_p(g, h_1)$ (use Hash-to-Point), and outputs $(\mathbb{G}, q, g, h_1, h_2)$ as the public parameters.

$(\mathsf{Add}, \mathsf{ASK}) \leftarrow \mathsf{FAPC.AddGen}(\mathsf{Par})$:

1. According to the public parameters $\mathsf{Par} = (\mathbb{G}, q, g, h_1, h_2)$, user Alice runs $(\mathsf{Add}, \mathsf{ASK}) \leftarrow \mathsf{TSLA.Gen}(\mathsf{Par})$ to get the long-term address of her account and the secret key, where $\mathsf{Add} = (A, B) = (g^{x_v}, g^{x_s})$ and $\mathsf{ASK} = (x_v, x_s)$;
2. Alice outputs Add, and retains ASK.

$(\mathsf{UTXO}, \mathsf{SK}_U, sig) \leftarrow \mathsf{FAPC.Mine}(a, \mathsf{Add}, \mathsf{ASK})$:

1. When a miner successfully gets the mining reward with amount a, he samples $x \leftarrow \mathbb{Z}_q^*$ and computes $c = g^x h_2^a$;
2. The miner uniformly generates $\mathsf{PK}_U, \mathsf{SK}_U$ for his UTXO;
3. The miner runs $\pi_{\mathsf{TRP}}(c) \leftarrow \mathsf{TRP.Prove}(c)$ to get the traceable range proof of c;
4. The miner computes $B \cdot c/h_2^a = g^{x+x_s}$, then sign with $x + x_s$ to get $sig \leftarrow \mathsf{Sign}_{x+x_s}(\mathsf{PK}_U, a, c, \pi_{\mathsf{TRP}}(c))$;
5. The miner outputs $(\mathsf{UTXO} = (\mathsf{PK}_U, c, \pi_{\mathsf{TRP}}(c)), sig)$ and retains SK_U.

$(L_{\mathsf{PK}}, L_{\mathsf{Add}}, \mathsf{UTXO}_{out}, \sigma_1, \sigma_2, \sigma_3) \leftarrow \mathsf{FAPC.Spend}(a_{in}, \mathsf{SK}_{in}, \mathsf{UTXO}_{in}, \mathsf{Add}_{out})$:

1. When Alice wants to pay Bob with her $\mathsf{UTXO}_{in} = (\mathsf{PK}_{in}, c_{in}, \pi_{\mathsf{TRP}}(c_{in}))$, she chooses another $l-1$ addresses, together with Bob's address Add_{out}, to generate a set of addresses L_{Add}, then Alice runs $\sigma_1 \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add}_{out})$, where $\sigma_1 = (\mathsf{PK}_{out}, ct, R, R_1, R_2, \theta)$;
2. Alice samples $x_{out} \leftarrow \mathbb{Z}_q^*$, computes the output commitment $c_{out} = g^{x_{out}} h^{a_{out}}$, runs $\pi_{\mathsf{TRP}}(c_{out}) \leftarrow \mathsf{TRP.Prove}(c_{out})$ to get the traceable range proof of c_{out}, she also generates a ciphertext $\sigma_3 \leftarrow Enc_{\mathsf{Add}}(a_{out}, x_{out})$;
3. Alice chooses another $m-1$ UTXOs, together with her UTXO_{in}, to generate a list of UTXO public key $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$, where $\mathsf{PK}_{in} \in L_{\mathsf{PK}}$;
4. Alice extracts the commitments $\{c_i\}_{i\in[1,m]}$ from $\{\mathsf{UTXO}_i\}_{i\in[1,m]}$, where $c_{in} \in \{c_i\}_{i\in[1,m]}$, generates $L_v = \{c_1/c_{out}, \cdots, c_m/c_{out}\}$, then runs $\sigma_2 \leftarrow \mathsf{TLRS.Sign}(L_{\mathsf{PK}}, L_v, \mathsf{UTXO}_{out}, \sigma_1, \sigma_3, \mathsf{SK}_{in})$ to get the traceable and linkable ring signature to hide the input UTXO_{in};
5. Alice outputs $(L_{\mathsf{PK}}, L_{\mathsf{Add}}, \mathsf{UTXO}_{out}, \sigma_1, \sigma_2, \sigma_3)$ as the transaction output.

$1/linked/0 \leftarrow \mathsf{FAPC.Verify}(L_{\mathsf{PK}}, L_{\mathsf{Add}}, \mathsf{UTXO}_{out}, \sigma_1, \sigma_2)$:

1. Verifier runs $\mathsf{TRLS.Link}(\sigma_2)$ to check whether double spending happens, if yes then he outputs *linked* and aborts, otherwise he continues to the next step;
2. Verifier runs $\mathsf{TRLS.Verify}(\sigma_2)$ to check the validity of TLRS ring signature;
3. Verifier runs $\mathsf{TSLA.Verify}(\sigma_1)$ to check the validity and correctness of PK_{out};
4. Verifier runs $\mathsf{TRP.Verify}(\pi_{\mathsf{TRP}}(c_{out}), c_{out})$ to check the validity of amount;
5. If all passed then outputs 1, otherwise outputs 0.

$(\mathsf{SK}_{out}, a_{out}^*, x_{out}^*) / \perp \leftarrow \mathsf{FAPC.Receive}(L_{\mathsf{Add}}, \mathsf{UTXO}_{out}, \sigma_1, \sigma_3, y, \mathsf{ASK})$:

1. The receiver Bob checks whether his address Add appears in L_{Add}, if yes he runs $\mathsf{SK}_U \leftarrow \mathsf{TSLA.Receive}(L_{\mathsf{Add}}, \sigma_1, \mathsf{ASK})$ to get the secret key SK_U of UTXO_{out}, if failed then outputs / $\perp$ and aborts;
2. Bob gets the amount and blinding element by decryption $(a_{out}^*, x_{out}^*) \leftarrow Dec_{\mathsf{ASK}}(\sigma_3)$;
3. Bob checks whether $g^{x_{out}^*} h_2^{a_{out}^*} \stackrel{?}{=} c_{out}$, if yes he receives UTXO_{out} to his wallet, otherwise aborts.

$(\kappa^*, \mathsf{Add}^*_{out}, a^*_{out}) / \perp \leftarrow \mathsf{FAPC.Audit}(L_{\mathsf{PK}}, L_{\mathsf{Add}}, \mathsf{UTXO}_{out}, \sigma_1, \sigma_2, y)$:

1. The auditor runs $\kappa^* / \perp \leftarrow \mathsf{TRLS.Trace}(\sigma_2, y)$ to recover the identity of input UTXO_{in} in L_{PK};
2. The auditor runs $\mathsf{Add}^* / \perp \leftarrow \mathsf{TSLA.Trace}(\sigma_1, y)$ to recover the long-term address of Bob;
3. The auditor runs $a^* / \perp \leftarrow \mathsf{TRP.Trace}(\pi_{\mathsf{TRP}}(c_{out}), y)$ to recover the output amount of transaction;
4. If $\perp$ appears in any step, then the auditor aborts, otherwise he outputs $(\kappa^*, \mathsf{Add}^*_{out}, a^*_{out})$.

Note that the transaction described in Algorithm 1 has one input UTXO and one output UTXO (with $a_{in} = a_{out}$), the solution for multi-input(output) transaction of FAPC is same as Monero, please refer to [26] for detailed description. Moreover, in FAPC, ring signature plays the key role in TLRS, TRP and TSLA, the choice of ring signature is not restricted, we can choose the most suited elliptic-curve-based ring signature (such as AOS, AOS'[1], Ring-CT 3.0[36]) to achieve better efficiency or compactness.

3.2 Security Requirements

The security requirements of FAPC contains anonymity, confidentiality, double-spending resistance and malicious auditor resistance. We introduce these requirements respectively:

1. **Anonymity** of FAPC consists of the anonymity of address and anonymity of money flow (UTXO), similar to Monero. The anonymity of address is provided by the anonymity of TSLA to protect the privacy of recipient address in transaction. The anonymity of money flow is provided by the anonymity of TLRS to protect the identity of input UTXO in L_{PK} so as to hide the money flow in the blockchain.
2. **Confidentiality** of FAPC is that the amount and blinding element (a, x) are hidden to others (excluding the auditor). The confidentiality of (a, x) is provided by the hiding property of Pedersen commitment and the zero-knowledge property of TRP, the verifier can only check the validity of $a \in [0, 2^n - 1]$ while cannot learn nothing else about (a, x).
3. **Double Spending Resistance** of FAPC is similar to Monero that any user cannot double spend his UTXOs in FAPC. Double spending resistance of FAPC is provided by the linkability of TLRS, in which the verifier can check whether double spending happens by usage of $\mathsf{TLRS.Link}$.
4. **Public Verifiability** of FAPC is that the correctness and validity of a transaction can be verified by the verifier, without any help from the auditor. FAPC can operate normally without the auditor.
5. **Full Auditability** of FAPC means that the auditor can trace the uses' addresses, identities of UTXOs (money flows) as well as the amounts in all transactions. The traceability of address is provided by the traceability

of TSLA, the traceability of money flow is provided by the traceability of TLRS, the traceability of transaction amount is provided by the traceability of TRP.

6. **Malicious Auditor Resistance** of FAPC is an enhanced security requirement which means that FAPC remains secure when the auditor is malicious or corrupted. For an adversary $\mathcal{A}$ with possession of the audit trapdoor, he cannot double spend, corrupt users, modify the value of UTXOs, or learn other privacy information including the UTXO secret key SK_U and the blinding element x (in $c = g^x h^a$). Note that the anonymity and confidentiality no longer hold for $\mathcal{A}$. It is a balance between privacy protection and auditability, and is a closer approach towards decentralization.

In the rest of this paper we give the detailed descriptions of TLRS, TRP and TSLA, with their security proofs against malicious auditors, to finish the construction of FAPC. We introduce the position-preserving multi-ring signature based on AOS and AOS' in the Appendix A.

4 Traceable and Linkable Ring Signature

In this section, we give the construction and security proofs of traceable and linkable ring signature scheme (TLRS). TLRS achieves unforgeability, anonymity, linkability, nonslanderability and traceability against malicious auditors. In the application of FAPC, unforgeability works for security of users' accounts, anonymity works for anonymity of the input UTXO, linkability and nonslanderability works for prevention of double-spending (actively or passively), traceability works for unconditional auditability of the input UTXO.

4.1 Construction

In our construction of TLRS, we use ring signature (AOS, AOS' or Ring-CT 3.0) as the ring signature component. Actually, we assume these schemes are anonymous and unforgeable in the random oracle model, which makes TLRS secure against malicious auditors, under standard assumptions. We give the introduction of TLRS in the following (single ring as example):

Algorithm 2: TLRS

$\mathsf{Par} \leftarrow \mathsf{TLRS.Setup}(\lambda)$:

1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and a generator $g \in \mathbb{G}$, the auditor generates $y \in \mathbb{Z}_q$ as the trapdoor, computes $h_1 = g^y$, system computes $h_2 = H_p(g, h_1)$ (use Hash-to-Point), and outputs $(\mathbb{G}, q, g, h_1, h_2)$ as the public parameters.

$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{TLRS.KeyGen}(\mathsf{Par})$:

1. According to the public parameters $(\mathbb{G}, q, g, h_1, h_2)$, user Alice samples $x \in \mathbb{Z}_q^*$ as her secret key, then computes $\mathsf{PK} = g^x$;
2. Alice outputs $\mathsf{PK} = g^x$, and retains $\mathsf{SK} = x$.

$\sigma \leftarrow \mathsf{TLRS.Sign}(\mathsf{SK}_\kappa, \mu, L_{\mathsf{PK}})$:

1. For a message μ, Alice chooses another $m-1$ users, together with her own public key, to generate a list of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$, where Alice's $\mathsf{PK} = \mathsf{PK}_\kappa \in L_{\mathsf{PK}}, \kappa \in \{1, \cdots, m\}$;
2. Alice outputs $\mathsf{TK} = h_1^{x_\kappa}$ and $I = h_2^{x_\kappa}$, then computes $e_1 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1)$, $e_2 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$;
3. Alice computes and outputs

$$L_{\mathsf{RPK}} = \{\mathsf{PK}_1 \cdot \mathsf{TK}^{e_1} \cdot I^{e_2}, \cdots, \mathsf{PK}_m \cdot \mathsf{TK}^{e_1} \cdot I^{e_2}\}$$

$$= \{g^{x_1} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}, \cdots, g^{x_m} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}\};$$

4. Alice runs ring signature $\tau \leftarrow \mathsf{Rsign}(\mathsf{SK}, \mu, L_{\mathsf{RPK}}, \mathsf{TK}, I)$ using L_{RPK} and $SK = x_\kappa$, outputs τ (use $gh_1^{e_1} h_2^{e_2}$ as the generator);
5. Alice outputs $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$.

$1/0 \leftarrow \mathsf{TLRS.Verify}(\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$:

1. Verifier computes $e_1^* = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1)$ and $e_2^* = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$;
2. Verifier computes $L_{\mathsf{RPK}}^* = \{\mathsf{PK}_1 \cdot \mathsf{TK}^{e_1^*} \cdot I^{e_2^*}, \cdots, \mathsf{PK}_m \cdot \mathsf{TK}^{e_1^*} \cdot I^{e_2^*}\}$;
3. Verifier checks the validity of ring signature τ ($gh_1^{e_1^*} h_2^{e_2^*}$ as the generator);
4. If all passed then outputs 1, otherwise outputs 0.

$linked/unlinked \leftarrow \mathsf{TLRS.Link}(\sigma, \sigma')$:

1. For two valid sTLRS' signatures $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$ and $\sigma' = (\tau', \mu', L'_{\mathsf{PK}}, \mathsf{TK}', I')$, if $I = I'$ then verifier outputs *linked*, otherwise outputs *unlinked*.

$\kappa^* / \perp \leftarrow \mathsf{TLRS.Trace}(\sigma, y)$:

1. For $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$, the auditor extracts $\mathsf{PK}_1, \cdots, \mathsf{PK}_m$ from L_{PK}, computes PK_i^y for $i = 1, \cdots, m$, outputs the smallest $\kappa^* \in \{1, \cdots, m\}$ such that $\mathsf{TK} = \mathsf{PK}_{\kappa^*}^y$ as the trace result, otherwise outputs $\perp$.

Correctness

Theorem 12 (Correctness of TLRS) *For an honest user Alice in TLRS, she can complete the traceable and linkable ring signature successfully, and the behavior of double signing (double spending) will be detected while the identity of Alice remaining anonymous. Moreover, the auditor can trace her identity correctly.*

Proof. In TLRS, for Alice's public key $\mathsf{PK} = \mathsf{PK}_\kappa = g^{x_\kappa}$, then Alice will output $\mathsf{TK} = h_1^{x_\kappa}$ and $I = h_2^{x_\kappa}$ with $L_{\mathsf{RPK}} = \{g^{x_1} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}, \cdots, g^{x_m} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}\}$. Since $\mathsf{RPK}_\kappa = g^{x_\kappa} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa} = (gh_1^{e_1} h_2^{e_2})^{x_\kappa}$, then Alice can use $\mathsf{SK} = x_\kappa$ to generate the ring signature τ using $gh_1^{e_1} h_2^{e_2}$ as the generator.

When double signing occurs, we know from the linkability of TLRS that Alice must have used $I = h_2^{x_\kappa}$ for twice (proved in Theorem 14), then the verifier can

detect that double signing occurs and outputs *linked*, at the same time, anyone (except for the auditor) cannot learn any information about the identity of signer by the anonymity of TLRS (proved in Theorem 13).

For the auditor, he can compute $\mathsf{PK}_\kappa^y = g^{yx_\kappa} = h_1^{x_\kappa} = \mathsf{TK}$ and then outputs $\mathsf{TLRS.Trace}(\sigma, y) = \kappa$ correctly. □

4.2 Security proofs

Proof of Anonymity

Theorem 13 (Anonymity of TLRS) *TLRS is anonymous for any PPT adversary $\mathcal{A}$ without possession of the trapdoor, assuming the ring signature is simulatable by programming the random oracle in the random oracle model.*

Proof. Assume $\mathcal{A}$ is playing the game with $\mathcal{S}$ in Definition 2, $\mathcal{A}$ generates a message μ and a list of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$, where $\mathsf{PK}_i = g^{x_i}$, and all PK_is are returned by $\mathcal{JO}$, and $\mathcal{S}$ knows all $SK_i = x_i$.

We consider the following games between $\mathcal{S}$ and $\mathcal{A}$:

- **Game 0**. $\mathcal{S}$ samples $\kappa \in \{1, \cdots, m\}$ uniformly at random, publishes $\mathsf{TK} = h_1^{x_\kappa}$ and $I = h_2^{x_\kappa}$, computes $e_1 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1)$, $e_2 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$ and $L_{\mathsf{RPK}} = \{g^{x_1} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}, \cdots, g^{x_m} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}\}$, generates the ring signature $\tau = \mathsf{Rsign}(SK, \mu, L_{\mathsf{RPK}}, \mathsf{TK}, I)$, outputs $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$ to $\mathcal{A}$. When $\mathcal{A}$ receives σ, he gives a guess $\kappa^* \in \{1, \cdots, m\}$.
- **Game 1**. $\mathcal{S}$ samples $\kappa \in \{1, \cdots, m\}, r_1, r_2 \in \mathbb{Z}_q^*$ uniformly at random, publishes $\mathsf{TK} = h_1^{r_1}$ and $I = h_2^{r_2}$, computes $e_1 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1)$, $e_2 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$ and $L_{\mathsf{RPK}} = \{g^{x_1} h_1^{e_1 r_1} h_2^{e_2 r_2}, \cdots, g^{x_m} h_1^{e_1 r_1} h_2^{e_2 r_2}\}$, generates the ring signature $\tau = \mathsf{Rsign}(\mu, L_{RK}, \mathsf{TK}, I)$ by programming the random oracle, outputs $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$ to $\mathcal{A}$. When $\mathcal{A}$ receives σ, he gives a guess $\kappa^* \in \{1, \cdots, m\}$.

In the two games above, Game 0 is the real game between $\mathcal{S}$ and $\mathcal{A}$ in TLRS, and Game 1 is the simulated game in the random oracle model. In game 1, κ is uniformly sampled by $\mathcal{S}$, which is statistical independent from the L_{PK}, then $\mathsf{Pr}_{\mathcal{A}}[\kappa^* = \kappa] = 1/m$.

Then we only need to prove that game 0 and game 1 are computational indistinguishable. If fact, the differences between the two games are the generations of TK, I and L_{RPK}. According to DDH assumption, $(g, h_i, g^{x_\kappa}, h_i^{x_\kappa})$ and $(g, h_i, g^{x_\kappa}, h_i^{r_i})$ are computational indistinguishable for $i = 1, 2$, then $\mathcal{A}$ cannot distinguish $h_i^{x_\kappa}$ (in game 0) from $h_i^{r_i}$ (in game 1). Then we know $\mathcal{A}$ cannot distinguish $\{g^{x_1} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}, \cdots, g^{x_m} h_1^{e_1 x_\kappa} h_2^{e_2 x_\kappa}\}$ from $\{g^{x_1} h_1^{e_1 r_1} h_2^{e_2 r_2}, \cdots, g^{x_m} h_1^{e_1 r_1} h_2^{e_2 r_2}\}$, then we know game 0 and game 1 are computational indistinguishable, then we finish the anonymity proof of TLRS. □

Proof of Linkability

Theorem 14 (Linkability of TLRS) *TLRS is linkable for any PPT adversary $\mathcal{A}$ (including the malicious auditor), assuming the unforgeability of ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ with possession of the trapdoor y, when $\mathcal{A}$ finished the link game with $\mathcal{S}$ in Definition 3, we assume that $\mathcal{A}$ wins the link game with nonnegligible advantage δ, that is, $\mathcal{A}$ returned k TLRS signatures $\sigma_i = (\tau_i, \mu_i, L^i_{\mathsf{PK}}, \mathsf{TK}_i, I_i), i = 1, \cdots, k$ (τ_is are the ring signatures), satisfying the following requirements:

1. All $\sigma_i, i = 1, \cdots, k$ are not returned by $\mathcal{SO}$.
2. All public keys from $L^i_{\mathsf{PK}}, i = 1, \cdots, k$ are returned by $\mathcal{JO}$.
3. $\mathsf{TLRS.Verify}(\sigma_i, L^i_{\mathsf{PK}}, \mu_i) = 1$ for $i = 1, \cdots, k$.
4. $\mathcal{A}$ queried $\mathcal{CO}$ less than k times.
5. $\mathsf{TLRS.Link}((\sigma_i, L^i_{\mathsf{PK}}, \mu_i), (\sigma_j, L^j_{\mathsf{PK}}, \mu_j)) = \textit{unlinked}$ for $i \neq j \in \{1, \cdots, k\}$.

We first prove a statement that, for a list of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$ returned by $\mathcal{JO}$ with $\mathsf{PK}_i = g^{x_i}$, any PPT adversary $\mathcal{A}$ generates a valid TLRS signature $\sigma \leftarrow\!\!\!\!\!- \mathcal{SO}$ if and only if he quires the $\mathcal{CO}$ at least once, except for negligible probability $\epsilon_0 = negl(\lambda)$.

- $\Rightarrow$. If $\mathcal{A}$ gets $\mathsf{SK} = x_i$ from $\mathcal{CO}$, and then $\mathcal{A}$ can run the TLRS signature scheme to generate a valid signature $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$.
- $\Leftarrow$. Assume $\mathcal{A}$ did not query the $\mathcal{CO}$ and $\mathcal{SO}$ for $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$ and finished the TLRS signature over $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$ with nonnegligible probability δ_1. We first prove that $\mathcal{A}$ does not know any of the secret keys in L_{PK}. Actually, under the hardness of discrete logarithm, $\mathcal{A}$ cannot compute x_i from $\mathsf{PK}_i = g^{x_i}, i = 1, \cdots, m$, then the probability of $\mathcal{A}$ obtaining any of x_i is $\epsilon_1 = negl(\lambda)$.
  Next, according to the assumption that $\mathcal{A}$ generates a valid signature $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$, then he must have finished the classic signature τ (with generator $gh_1^{e_1}h_2^{e_2}$), where $e_1 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1)$, $e_2 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$. Without loss of generality, we assume $\mathsf{TK} = g^{s_1}h_2^{t_1}$ and $I = g^{s_2}h_2^{t_2}$ output by $\mathcal{A}$ (assume $\mathcal{A}$ knows y), then we have

$$\begin{aligned} L_{\mathsf{RPK}} &= \{g^{x_1}(g^{s_1}h_2^{t_1})^{e_1}(g^{s_2}h_2^{t_2})^{e_2}, \cdots, g^{x_m}(g^{s_1}h_2^{t_1})^{e_1}(g^{s_2}h_2^{t_2})^{e_2}\} \\ &= \{g^{x_1+s_1e_1+s_2e_2}h_2^{t_1e_1+t_2e_2}, \cdots, g^{x_m+s_1e_1+s_2e_2}h_2^{t_1e_1+t_2e_2}\}. \end{aligned}$$

  Since the ring signature scheme achieves unforgeability, and $\mathcal{A}$ finished the ring signature τ with L_{RPK} under generator $gh_1^{e_1}h_2^{e_2}$, then we get $\mathcal{A}$ knows $\mathsf{SK} = z$ for at least one $i \in \{1, \cdots, m\}$ *s.t.* $g^{x_1+s_1e_1+s_2e_2}h_2^{t_1e_1+t_2e_2} = (gh_1^{e_1}h_2^{e_2})^z$, except for negligible probability $\epsilon_2 = negl(\lambda)$. We can also assume that $e_1 = 0$ or $e_2 = 0$ happens with negligible probability $\epsilon_3 = negl(\lambda)$, which means $\mathcal{A}$ gets a relation for $g^{x_i - z + e_1(s_1 - yz) + e_2s_2} = h_2^{e_2(z-t_2) - t_1e_1}$ with nonnegligible probability $\delta_1 - \epsilon_1 - \epsilon_2 - \epsilon_3$, if $e_2(z - t_2) - t_1e_1 \neq 0$, then the relation is nontrivial, which contradicts with the hardness of discrete logarithm problems, so we have $e_2(z - t_2) = t_1e_1$, if $t_1 \neq 0$, then $e_1 = e_2(z - t_2)t_1^{-1}$, which

means e_1 is pre-computed before $\mathcal{A}$ runs the hash function (random oracle), which happens with negligible probability $\epsilon_4 = negl(\lambda)$, then we get $t_1 = z - t_2 = 0$. Then we have $x_i - z + e_1(s_1 - yz) + e_2 s_2 = 0$, from similar arguments we can get $s_2 = 0, z = t_2 = x_i, s_1 = yx_i$, then $\mathsf{TK} = h_1^{x_i}$ and $I = h_2^{x_i}$ with nonnegligible probability $\delta_1 - \epsilon_1 - \epsilon_2 - \epsilon_3 - \epsilon_4$, which contradicts to the assumptions above. Then we get that $\mathcal{A}$ generates a valid TLRS signature $\sigma \nleftarrow \mathcal{SO}$ if and only if he quires the $\mathcal{CO}$ at least once, except for negligible probability.

According to the fourth requirement that the number of times of $\mathcal{A}$ querying $\mathcal{CO}$ is $\leq k-1$, and $\mathcal{A}$ returned k valid TLRS signatures $\sigma_i = (\tau_i, \mu_i, L^i_{\mathsf{PK}}, \mathsf{TK}_i, I_i)$, $i = 1, \cdots, k$, then we know there are two TLRS signatures from the same query of $\mathcal{CO}$, saying $\mathsf{SK} = z$ from $\mathsf{PK} = g^z$, and $\mathcal{A}$ finished two unlinked valid TLRS signatures, then there is at least one $I_i = g^{s_2} h_2^{t_2} \neq h_2^z$ from the two TLRS signatures (otherwise they will be linked). We also set $\mathsf{TK} = g^{s_1} h_2^{t_1}$, then we have $L_{\mathsf{RPK}} = \{g^{x_1 + s_1 e_1 + s_2 e_2} h_2^{t_1 e_1 + t_2 e_2}, \cdots, g^{x_m + s_1 e_1 + s_2 e_2} h_2^{t_1 e_1 + t_2 e_2}\}$, since $\exists j \in \{1, \cdots, n\}$ $s.t.$ $\mathcal{A}$ have queried $\mathcal{CO}$ with input PK_j and signs with RPK_j, satisfying $z = x_j$ and $g^{s_2} h_2^{t_2} \neq h_2^z$, then we have there exists $w \in \mathbb{Z}_q^*$ $s.t.$ $\mathsf{RPK}_j = (g h_1^{e_1} h_2^{e_2})^w$ by the unforgeability of ring signature, except for negligible probability $\epsilon_5 = negl(\lambda)$, then we have $g^{z - w + e_1(s_1 - yw) + e_2 s_2} = h_2^{e_2(w - t_2) - t_1 e_1}$, if $e_2(w - t_2) - t_1 e_1 \neq 0$, then we get a nontrivial relation between g and h_2, which happens with negligible probability $\epsilon_6 = negl(\lambda)$ from the hardness of discrete logarithm. Then then we have $t_2 = w$ and $t_1 = 0$, otherwise e_1 (or e_2) is pre-computed before $\mathcal{A}$ runs the hash function (random oracle), which happens with negligible probability $\epsilon_7 = negl(\lambda)$. Then we get $z - w + e_1(s_1 - yw) + e_2 s_2 = 0$, from similar arguments we can get $s_2 = 0, z = w = t_2 = x_j, s_1 = yw$, except for negligible probability $\epsilon_8 = negl(\lambda)$. Then we know that $g^{s_2} h_2^{t_2} \neq h_2^z$ with nonnegligible probability $\delta - k\epsilon_0 - \epsilon_5 - \epsilon_6 - \epsilon_7$, this contradicts to the assumptions before, then we finish the linkability proof of TLRS. □

Proof of Nonslanderability

Theorem 15 (Nonslanderability of TLRS) *TLRS is nonslanderable for any PPT adversary $\mathcal{A}$ (including the malicious auditor), assuming the unforgeability of ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ with possession of the trapdoor y, when $\mathcal{A}$ finished the slandering game with $\mathcal{S}$ in Definition 4, $\mathcal{A}$ gave a list of public keys L_{PK}, a message μ and a public key $\mathsf{PK}_\kappa \in L_{\mathsf{PK}}$ to $\mathcal{S}$, $\mathcal{S}$ returns the corresponding signature $\sigma \leftarrow \mathsf{TLRS.Sign}(SK_\kappa, L_{\mathsf{PK}}, \mu)$ to $\mathcal{A}$. We assume that $\mathcal{A}$ wins the slandering game with nonnegligible advantage δ, that is, $\mathcal{A}$ successfully outputs a ring signature $\sigma^* = (\tau^*, \mu^*, L^*_{\mathsf{PK}}, \mathsf{TK}^*, I^*)$, satisfying the following:

1. $\mathsf{TLRS.Verify}(\sigma^*, L^*_{\mathsf{PK}}, \mu^*) = 1$.
2. PK_κ is not queried by $\mathcal{A}$ to $\mathcal{CO}$.
3. PK_κ is not queried by $\mathcal{A}$ as input to $\mathcal{SO}$.
4. $\mathsf{TLRS.Link}((\sigma, L_{\mathsf{PK}}, \mu), (\sigma^*, L^*_{\mathsf{PK}}, \mu^*)) = linked$.

From the definition of Link, we know that $I^* = I = h_2^{x_\kappa}$, since $\mathsf{PK}_\kappa = g^{x_\kappa}$ was not queried by $\mathcal{A}$ to $\mathcal{CO}$ and $\mathcal{SO}$, then $\mathcal{A}$ does not know $SK = x_\kappa$ except for negligible probability $\epsilon_0 = negl(\lambda)$ under the hardness of discrete logarithm problems. Then we know $\mathcal{A}$ successfully produced a valid ring signature τ^* with nonnegligible advantage $\delta - \epsilon_0$. According to the unforgeability of ring signature, we know that $\mathcal{A}$ knows at least one signing key except for negligible probability $\epsilon_1 = negl(\lambda)$, that is, there exists $j \in \{1, \cdots, m\}$, $\mathcal{A}$ knows x $s.t.$ $\mathsf{PK}_j^* \cdot (\mathsf{TK}^*)^{e_1} I^{e_2} = (g h_1^{e_1} h_2^{e_2})^x$ with nonnegligible advantage $\delta - \epsilon_0 - \epsilon_1$, where $e_1 = H(L_{\mathsf{PK}}^*, \mathsf{TK}^*, I, 1)$, $e_2 = H(L_{\mathsf{PK}}^*, \mathsf{TK}^*, I, 2)$. Without loss of generality, we assume $\mathsf{PK}_j^* = g^{s_1} h_2^{t_1}$, $\mathsf{TK}^* = g^{s_2} h_2^{t_2}$ output by $\mathcal{A}$, then we have $(g^{s_1} h_2^{t_1})(g^{s_2} h_2^{t_2})^{e_1} h_2^{e_2 x_\kappa} = (g h_1^{e_1} h_2^{e_2})^x$, then $g^{s_1 - x + e_1(s_2 - xy)} = h_2^{e_2(x - x_\kappa) - t_2 e_1 - t_1}$, using similar arguments in Theorem 14, we can get $s_1 = x = x_\kappa, s_2 = xy, t_1 = t_2 = 0$ with nonnegligible probability, this means $\mathsf{PK}_\kappa = g^{x_\kappa}$ was queried by $\mathcal{A}$ to $\mathcal{CO}$ to get $s_1 = x = x_\kappa$, which contradicts to the assumptions before, then we finish the nonslanderability proof of TLRS. □

According to lemma 5, we get the unforgeability of TLRS:

Corollary 16 (Unforgeability) *TLRS is unforgeable for any PPT adversary $\mathcal{A}$, including the malicious auditor.*

Proof of Traceability

Theorem 17 (Traceability of TLRS) *TLRS is traceable for any PPT adversary $\mathcal{A}$ (including the malicious auditor), assuming the unforgeability of ring signature component.*

Proof. For a PPT adversary $\mathcal{A}$ with possession of the trapdoor y, when $\mathcal{A}$ finished the tracing game with $\mathcal{S}$ in Definition 6, $\mathcal{A}$ generates a list of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_m\}$, we assume that $\mathcal{A}$ wins the tracing game with nonnegligible advantage δ, that is, $\mathcal{A}$ generates a TLRS signature $\sigma = (\tau, \mu, L_{\mathsf{PK}}, \mathsf{TK}, I)$ using $\mathsf{PK}_\kappa \in L_{\mathsf{PK}}$, satisfying the following:

1. $\mathsf{TLRS.Verify}(\sigma, L_{\mathsf{PK}}, \mu) = 1$.
2. $\mathsf{PK}_i \neq \mathsf{PK}_j$ for $1 \leq i < j \leq m$.
3. $\mathsf{TLRS.Trace}(\sigma, y) \neq \kappa$ or $\mathsf{Trace}(\sigma, y) = \perp$.

We assume $\mathsf{PK}_i = g^{x_i} h_2^{y_i}, i = 1, \cdots, m$ returned by $\mathcal{A}$ without loss of generality, and assume $\mathsf{TK} = g^{s_1} h_2^{t_1}, I = g^{s_2} h_2^{t_2}$. Then we have:

$$\begin{aligned} L_{\mathsf{RPK}} &= \{g^{x_1} h_2^{y_1} \cdot (g^{s_1} h_2^{t_1})^{e_1} (g^{s_2} h_2^{t_2})^{e_2}, \cdots, g^{x_m} h_2^{y_m} \cdot (g^{s_1} h_2^{t_1})^{e_1} (g^{s_2} h_2^{t_2})^{e_2}\} \\ &= \{g^{x_1 + s_1 e_1 + s_2 e_2} h_2^{y_1 + t_1 e_1 + t_2 e_2}, \cdots, g^{x_m + s_1 e_1 + s_2 e_2} h_2^{y_m + t_1 e_1 + t_2 e_2}\}. \end{aligned}$$

Where $e_1 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 1), e_2 = H(L_{\mathsf{PK}}, \mathsf{TK}, I, 2)$, moreover, we assume $e_i \neq 0$ for $i = 1, 2$, except for negligible probability $\epsilon_0 = negl(\lambda)$. According to the condition that $\mathcal{A}$ signed τ with PK_κ, then we get $\mathcal{A}$ knows the corresponding $\mathsf{SK}_\kappa = z$, except for negligible probability $\epsilon_1 = negl(\lambda)$, under the unforgeability of ring

signature. Then we have $g^{x_\kappa+s_1e_1+s_2e_2}h_2^{y_\kappa+t_1e_1+t_2e_2} = (gh_1^{e_1}h_2^{e_2})^z$ with nonnegligible probability $\delta-\epsilon_0-\epsilon_1$. Then $g^{x_\kappa-z+e_1(s_1-yz)+s_2e_2} = h_2^{e_2(z-t_2)-t_1e_1-y_\kappa}$, from the similar arguments in Theorem 14, we can get $z = t_2 = x_\kappa, s_1 = yz, s_2 = t_1 = 0$, otherwise $\mathcal{A}$ will get a non-trivial relationship between g and h_2, or e_i will be pre-determined before $\mathcal{A}$ runs the hash function (random oracle), both of which happen with negligible probability ϵ_2 according to the hardness of discrete logarithm problems and unpredictable of hash functions. Then we have $\mathsf{PK}_\kappa = g^{x_\kappa}$ and $\mathsf{PK}_\kappa^y = g^{x_\kappa y} = h_1^{x_\kappa} = \mathsf{TK}$, then $\mathsf{Trace}(\sigma, y) = \kappa$, which contradicts with the assumptions before, then we finish the traceability proof of TLRS. □

5 Traceable Range Proofs

In this section we give the construction and security proof of traceable range proof (TRP), which provides the validity ($\in [0, 2^n-1]$), confidentiality and traceability of the amounts in all FAPC transactions by modifying the Borromean range proof with extra tracing keys and tags to achieve security against malicious auditors. In the application of FAPC, soundness works for the correctness and security of UTXO amount, zero-knowledge works for confidentiality of the UTXO amount, traceability works for unconditional auditability of the UTXO amount.

5.1 Construction of Traceable Range Proof

In the construction of TRP, we use Pedersen commitment and bit expansion of amount, then add tracing keys bitwise into the proof, the sets of public keys for Borromean ring signature is also modified, compared to Borromean range proof. There exists an auditor who can use the trapdoor and tracing keys to recover the hidden amount of UTXO. We give the introduction of TRP in the following:

Algorithm 3: TRP

$\mathsf{Par} \leftarrow \mathsf{TRP.Setup}(\lambda)$:

1. System chooses an elliptic curve $\mathbb{G}$ with prime order q and samples a generator $g \leftarrow \mathbb{G}$. The auditor generates $y \in \mathbb{Z}_q^*$ as the trapdoor, computes $h_1 = g^y$, system computes $h_2 = H_p(g, h_1)$ (use Hash-to-Point), and outputs $(\mathbb{G}, q, g, h_1, h_2)$ as the public parameters.

$\pi_{\mathsf{TRP}}(c) \leftarrow \mathsf{TRP.Prove}(\mathsf{Par}, a)$:

1. According to the public parameters and amount $a \in [0, 2^n - 1]$, prover Alice samples $x \in \mathbb{Z}_q^*$ uniformly, computes $c = g^x h_2^a$ as the commitment;
2. Alice computes the binary expansion $a = a_0 + \cdots + 2^{n-1}a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$, samples $x_0, \cdots, x_{n-1}$ uniformly, computes $\beta = x - x_0 - \cdots - x_{n-1}$;

3. For every $i = 0, \cdots, n-1$, Alice computes $c_i = g^{x_i} h_2^{2^i a_i}, c_i' = g^{x_i} h_2^{2^i a_i - 2^i}, I_i = h_2^{x_i}, \mathsf{TK}_i = h_1^{x_i}$, outputs $\{c_i, c_i', \mathsf{TK}_i, I_i\}_{i \in [0,n-1]}$;
4. Alice computes $e_k = H(\{c_i, \mathsf{TK}_i, I_i\}_{i \in [0,n-1]}, k)$ for $k = 1, 2$;
5. Alice computes $L_{\mathsf{PK}}^i = \{c_i \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}, c_i' \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}\}$ for $i = 0, \cdots, n-1$;
6. Alice runs the Borromean multi-ring signature with $L_{\mathsf{PK}} = \{L_{\mathsf{PK}}^0, \cdots, L_{\mathsf{PK}}^{n-1}\}$, computes $\sigma \leftarrow \mathsf{Rsign}(\mathsf{SK}, \mu, L_{\mathsf{PK}})$ (use generator $g h_1^{e_1} h_2^{e_2}$ in each ring) for message $\mu = (c, L_{\mathsf{PK}})$;
7. Alice outputs the TRP proof $\pi_{\mathsf{TRP}}(c) = (c, \beta, \{c_i, c_i', \mathsf{TK}_i, I_i\}_{i \in [0,n-1]}, \sigma)$.

$1/0 \leftarrow \mathsf{TRP.Verify}(\pi_{\mathsf{TRP}}(c))$:

1. For every $i = 0, \cdots, n-1$, verifier checks $c_i / c_i' \stackrel{?}{=} h_2^{2^i}$ and $g^\beta \cdot \prod_{i=0}^{n-1} c_i \stackrel{?}{=} c$;
2. Verifier computes $e_k^* = H(\{c_i, \mathsf{TK}_i, I_i\}_{i \in [0,n-1]}, k)$ for $k = 1, 2$;
3. Verifier computes $L_{\mathsf{PK}}^{i^*} = \{c_i \cdot \mathsf{TK}_i^{e_1^*} \cdot I_i^{e_2^*}, c_i' \cdot \mathsf{TK}_i^{e_1^*} \cdot I_i^{e_2^*}\}$ for $i = 0, \cdots, n-1$;
4. Verifier checks the validity of Borromean multi-ring signature σ for n rings $\{L_{\mathsf{PK}}^{i^*}\}_{i=0,\cdots,n-1}$ (use generator $g h_1^{e_1^*} h_2^{e_2^*}$ in each ring), if all passed then outputs 1, otherwise outputs 0.

$a^* \leftarrow \mathsf{TRP.Trace}(y, \pi_{\mathsf{TRP}}(c))$:

1. For every $i = 0, \cdots, n-1$, the auditor computes c_i^y;
2. For every $i = 0, \cdots, n-1$, if $c_i^y = \mathsf{TK}_i$ then outputs $a_i^* = 0$, otherwise outputs $a_i^* = 1$;
3. The auditor outputs $a^* = a_0^* + \cdots + 2^{n-1} a_{n-1}^*$ as the tracing result.

Proof of Correctness

Theorem 18 (Correctness of TRP) *For an honest prover Alice, she can run TRP successfully and the auditor can trace the hidden amount correctly.*

Proof. According to the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}$ of a and $c_i = g^{x_i} h_2^{2^i a_i}, c_i' = g^{x_i} h_2^{2^i a_i - 2^i}$, we know there is only one element being a power of $g \cdot h_1^{e_1} \cdot h_2^{e_2}$ in $L_{\mathsf{PK}}^i = \{c_i \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}, c_i' \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}\}$ known by Alice, then Alice can use the secret keys $\mathsf{SK} = (x_0, \cdots, x_{n-1})$ ($\mathsf{SK}_i = x_i$ with corresponding $\mathsf{PK}_i = (g \cdot h_1^{e_1} \cdot h_2^{e_2})^{x_i}$) to finish the Borromean multi-ring signature for $L_{\mathsf{PK}} = \{L_{\mathsf{PK}}^0, \cdots, L_{\mathsf{PK}}^{n-1}\}$. Besides, we know that $g^\beta \cdot \prod c_i = c$ and $c_i / c_i' = h_2^{2^i}$ from the $\mathsf{TRP.Gen}$ algorithm. When $a_i = 0$, we know $c_i = g^{x_i} h_2^{2^i a_i} = g^{x_i}$, $\mathsf{TK}_i = h_1^{x_i} = c_i^y$. When $a_i = 1$, we know $\mathsf{TK}_i = h_1^{x_i}, c_i^y = (g^{x_i} h_2^{2^i})^y = h_1^{x_i} h_2^{2^i y}$, then $c_i^y = \mathsf{TK}_i$ iff $y = 0$, which happens with negligible probability, then the auditor can recover a correctly, so we get the correctness of TRP. □

Proof of Soundness Completeness is easily obtained from the correctness of TRP, here we prove the soundness of TRP, which means any PPT adversary with possession of all trapdoors cannot generate a valid $\pi_{\mathsf{TRP}}(c)$ for $c = g^x h_2^a$ and $a \notin [0, 2^n - 1]$, under the hardness of discrete logarithm and the unforgeability of Borromean multi-ring signature.

Theorem 19 (Soundness of TRP) *TRP has computational soundness for any PPT adversary $\mathcal{A}$, including the malicious auditor.*

Proof. For $c = g^x h_2^a$ with $a \notin [0, 2^n - 1]$, assume $\mathcal{A}$ (with possession of the trapdoor) successfully outputs a valid proof $\pi_{\mathsf{TRP}}(c)$, then we have $g^\beta \cdot \prod_{i=0}^{n-1} c_i = c$ and $c_i/c_i' = h_2^{2^i}$. Without loss of generality, we set $c_i = g^{x_i} h_2^{b_i}$, then $c_i' = g^{x_i} h_2^{b_i - 2^i}$, since $a \notin [0, 2^n - 1]$, we know there exists at least one $l \in \{0, \cdots, n-1\}$, satisfying $b_l \neq 0$ and $b_l \neq 2^l$, otherwise $g^\beta \cdot \prod_{i=0}^{n-1} c_i = g^{\beta + \sum x_i} h_2^{\sum b_i} = g^x h_2^a$ with $\sum b_i \in [0, 2^n - 1]$ implies a nontrivial relation $h_2^{a - \sum b_i} = g^{\beta + \sum x_i - x}$ between g and h, which happens with negligible probability $\epsilon_0 = negl(\lambda)$. Then we have:

$$L_{\mathsf{PK}}^l = \{c_l \cdot \mathsf{TK}_l^{e_1} \cdot I_l^{e_2}, c_l' \cdot \mathsf{TK}_l^{e_1} \cdot I_l^{e_2}\},$$

where $e_k = H(\{c_i, \mathsf{TK}_i, I_i\}_{i \in [0,n-1]}, k)$ for $k = 1, 2$, since $\mathcal{A}$ knows the trapdoor y, we can set $\mathsf{TK}_l = g^{s_1} h_2^{t_1}, I_l = g^{s_2} h_2^{t_2}$, then $L_{\mathsf{PK}}^l = \{g^{x_l + s_1 e_1 + s_2 e_2} h_2^{b_l + t_1 e_1 + t_2 e_2}$, $g^{x_l + s_1 e_1 + s_2 e_2} h_2^{b_l - 2^l + t_1 e_1 + t_2 e_2}\} = \{\mathsf{PK}_l, \mathsf{PK}_l'\}$ with $b_l \neq 0$ and $b_l \neq 2^l$. Since the generator for Borromean multi-ring signature is $g \cdot h_1^{e_1} \cdot h_2^{e_2} = g^{1 + y e_1} h_2^{e_2}$, from the unforgeability of Borromean ring signature, we know there is at least one public key from $\{\mathsf{PK}_l, \mathsf{PK}_l'\}$ (without loss of generality we set PK_l) satisfying $g^{x_l + s_1 e_1 + s_2 e_2} h_2^{b_l + t_1 e_1 + t_2 e_2} = (g^{1 + y e_1} h_2^{e_2})^z$, with z known to $\mathcal{A}$. Then we have $g^{x_l - z + e_1(s_1 - yz) + s_2 e_2} = h_2^{e_2(z - t_2) - t_1 e_1 - b_l}$. From similar arguments in Theorem 14, we can get $z = x_l = t_2, s_1 = yz, s_2 = 0, t_1 = 0, b_l = 0$, otherwise $\mathcal{A}$ will get a non-trivial relationship between g and h, or e_i will be pre-determined before $\mathcal{A}$ runs the hash function (random oracle), both of which happen with negligible probability ϵ_1. Then $b_l = 0$, which contradicts with the assumptions before, then we get the soundness of TRP against malicious auditors. □

Proof of Zero-knowledge

Theorem 20 (Zero-knowledge of TRP) *TRP is computational zero-knowledge for any PPT adversary $\mathcal{A}$ (without possession of the trapdoor).*

Proof. For every $i = 0, \cdots, n-1$, we consider the effect that $\{\mathsf{TK}_i, I_i\}_{i \in [0,n-1]}$ being added into the proof, and prove that $(c_i, \mathsf{TK}_i, I_i)$ is computational indistinguishable from uniform distribution when $a_i = 0$ or 1. Formally, we prove for $c_i = g^{x_i} h_2^{2^i a_i}, c_i' = g^{x_i} h_2^{2^i a_i - 2^i}$ with $c_i/c_i' = h_2^{2^i}$ being a constant, any PPT adversary $\mathcal{A}$ cannot distinguish uniform distribution $U = (r, r_1, r_2)$ from $(c_i, \mathsf{TK}_i, I_i) = (g^{x_i}, h_1^{x_i}, h_2^{x_i})$ (when $a_i = 0$) or $(c_i, \mathsf{TK}_i, I_i) = (g^{x_i} h_2^{2^i}, h_1^{x_i}, h_2^{x_i})$ (when $a_i = 1$), where U is sampled uniformly from $\mathbb{G}^3$.

Actually, we know that $(g^{x_i}, h_1^{x_i}, h_2^{x_i})$ and (g^{x_i}, r_1, r_2) are computational indistinguishable for uniformly generated $x_i \in \mathbb{Z}_q^*$, under the extended DDH assumption. For g being a generator of $\mathbb{G}$, the distribution of (g^{x_i}, r_1, r_2) and (r, r_1, r_2) are identical. Let constant $u = h_2^{2^i}$, we know that the distribution of

(r, r_1, r_2) and (ru, r_1, r_2) are identical. Again from the extended DDH assumption, we know (ru, r_1, r_2) and $(g^{x_i}u, h_1^{x_i}, h_2^{x_i})$ are computational indistinguishable. Then we have the following relations:

$$(g^{x_i}, h_1^{x_i}, h_2^{x_i}) \approx_c (r, r_1, r_2) = U = (ru, r_1, r_2) \approx_c (g^{x_i}u, h_1^{x_i}, h_2^{x_i}).$$

Where $g, h, h_1, u \in \mathbb{G}$ are constants, $U \leftarrow \mathbb{G}^3, x_i \leftarrow \mathbb{Z}_q^*$ are sampled uniformly.

Since $(g^{x_i}, h_1^{x_i}, h_2^{x_i}) = (c_i, \mathsf{TK}_i, I_i)_{a_i=0}$ and $(g^{x_i}u, h_1^{x_i}, h_2^{x_i}) = (c_i, \mathsf{TK}_i, I_i)_{a_i=1}$, we know they are all computational indistinguishable from $U = (r, r_1, r_2)$ for any PPT adversary $\mathcal{A}$ without possession of trapdoors. Since all x_is are uniformly generated independently for every $i = 0, \cdots, n-1$, then we finish the zero-knowledge proof of TRP. □

Proof of Traceability

Theorem 21 (Traceability of TRP) *TRP is traceable for any PPT adversary $\mathcal{A}$ (including the malicious auditor).*

Proof is given in the Appendix B.1.

6 Tracing Scheme for Long-term Addresses

In this section we introduce TSLA, a tracing scheme for long-term address with security against malicious auditors, by usage of ring signature and ElGamal encryption. The address generation algorithm of TSLA is similar to Monero, while the key generation algorithm for every UTXO is modified to achieve both anonymity and traceability. In a transaction, the initiator chooses another $l-1$ users' addresses, together with the real recipient's address, to generate an address list $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$, then he encrypts the secret information and generates a double-ring signature to prove the validity of the ciphertext. The recipient can find his transaction and recover the secret key of the new UTXO. The auditor can trace the recipient's address by decryption. For a malicious auditor, he cannot make a valid transaction to escape from audit.

Construction

Algorithm 4: TSLA

$\mathsf{Par} \leftarrow \mathsf{TSLA.Setup}(\lambda)$:

1. System chooses an elliptic curve $\mathbb{G}$ and a generator $g \in \mathbb{G}$, the auditor generates $y \in \mathbb{Z}_q^*$ as the trapdoor, computes $h_1 = g^y$, system computes $h_2 = H_p(g, h_1)$ (hash to point), then outputs $(\mathbb{G}, q, g, h_1, h_2)$ as the public parameters.

$(\mathsf{Add}, \mathsf{ASK}) \leftarrow \mathsf{TSLA.Gen}(\mathsf{Par})$:

1. According to the public parameters $(\mathbb{G}, q, g, h_1, h_2)$, a user Bob samples $x_v, x_s \in \mathbb{Z}_q^*$ uniformly at random, computes $\mathsf{Add} = (A, B) = (g^{x_v}, g^{x_s})$;
2. Bob outputs his address $\mathsf{Add} = (A, B)$, and retains $\mathsf{ASK} = (x_v, x_s)$ as his secret keys.

$(L_{\mathsf{Add}}, \mathsf{PK}_{out}, ct, R, R_1, R_2, \theta) \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add})$:

1. For an initiator Alice who wants to pay Bob with her money UTXO_{in}, she chooses another $l-1$ users, together with Bob's address $\mathsf{Add} = (A, B)$, to generate an address list $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$, where Bob's $\mathsf{Add} = \mathsf{Add}_\kappa \in L_{\mathsf{Add}}, \kappa \in \{1, \cdots, l\}$ and $\mathsf{Add}_i = (A_i, B_i)$ for $i = 1, \cdots, l$;
2. Alice samples $z \in \mathbb{Z}_q^*$ uniformly at random, computes $R = A_\kappa^z, R_1 = h_1^z, R_2 = h_2^z$, then she computes $\mathsf{PK}_{out} = g^z \cdot B_\kappa$;
3. Alice computes $e_i = H(L_{\mathsf{Add}}, R, R_1, R_2, \mathsf{PK}_{out}, i)$ for $i = 1, 2$;
4. Alice computes $L_1 = \{\mathsf{PK}_{out} \cdot B_1^{-1} \cdot R_1^{e_1} \cdot R_2^{e_2}, \cdots, \mathsf{PK}_{out} \cdot B_l^{-1} \cdot R_1^{e_1} \cdot R_2^{e_2}\}$ and $L_2 = \{A_1 \cdot h_1^{e_1} \cdot h_2^{e_2}, \cdots, A_l \cdot h_1^{e_1} \cdot h_2^{e_2}\}$;
5. Alice runs the position preserving double-ring signature for L_1 and L_2, using g_1 (for L_1) and g_2 (for L_2) as the generator for each ring separately, where $g_1 = g \cdot h_1^{e_1} \cdot h_2^{e_2}$ and $g_2 = R \cdot R_1^{e_1} \cdot R_2^{e_2}$. Alice gets $\theta \leftarrow \mathsf{Rsign}(L_{\mathsf{Add}}, L_1, L_2, g_1, g_2, z)$;
6. Alice encrypts $ct = \mathsf{Enc}_{B_\kappa}(z)$, using the spending public key B_κ of Bob;
7. Alice outputs $(L_{\mathsf{Add}}, \mathsf{PK}_{out}, ct, R, R_1, R_2, \theta)$ as the output.

$1/0 \leftarrow \mathsf{TSLA.Verify}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, ct, R, R_1, R_2, \theta)$:

1. According to the public parameters $(\mathbb{G}, q, g, h_1, h_2)$, the verifier computes $e_i^* = H(L_{\mathsf{Add}}, R, R_1, R_2, PK_{out}, i)$ for $i = 1, 2$;
2. Verifier computes $L_1^* = \{\mathsf{PK}_{out} \cdot B_1^{-1} \cdot R_1^{e_1^*} \cdot R_2^{e_2^*}, \cdots, \mathsf{PK}_{out} \cdot B_l^{-1} \cdot R_1^{e_1^*} \cdot R_2^{e_2^*}\}$ and $L_2^* = \{A_1 \cdot h_1^{e_1^*} \cdot h_2^{e_2^*}, \cdots, A_l \cdot h_1^{e_1^*} \cdot h_2^{e_2^*}\}$;
3. Verifier checks the validity of θ with L_1^* and L_2^*, by using $g_1^* = g \cdot h_1^{e_1^*} \cdot h_2^{e_2^*}$ and $g_2^* = R \cdot R_1^{e_1^*} \cdot R_2^{e_2^*}$ as the generator for each ring separately;
4. If all passed then outputs 1, otherwise outputs 0.

$\mathsf{SK}_{out} / \perp \leftarrow \mathsf{TSLA.Receive}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, ct, R, \mathsf{ASK})$:

1. Bob checks whether his address appears in L_{Add}, if yes, he computes $B^* = \mathsf{PK}_{out} / R^{x_v^{-1}}$;
2. Bob checks whether $B_\kappa \stackrel{?}{=} B^*$, if yes, he recovers $z^* \leftarrow \mathsf{Dec}_{x_s}(ct)$;
3. Bob checks whether $g^{z^*} \cdot B_\kappa \stackrel{?}{=} \mathsf{PK}_{out}$, if yes, he computes $\mathsf{SK}_{out} = z^* + x_s$ as the secret key of the new UTXO, otherwise he outputs $\perp$.

$\mathsf{Add}_{\kappa^*} / \perp \leftarrow \mathsf{TSLA.Trace}(L_{\mathsf{Add}} \mathsf{PK}_{out}, R_1, y)$:

1. The auditor computes $B' = \mathsf{PK}_{out} / R_1^{y^{-1}}$;
2. The auditor searches the smallest $\kappa^* \in \{1, \cdots, l\}$ such that $B_{\kappa^*} = B'$, then he outputs Add_{κ^*} as the tracing result, otherwise he outputs $\perp$.

Note that the UTXO public key generation algorithm in TSLA is completely different from Monero, the TSLA provides $1/l$ anonymity of address, while Monero provides $1/M$ anonymity to hide the recipient's address in all users' addresses, where M is the number of all addresses in the blockchain. So the anonymity of Monero is stronger than FAPC. Moreover, for the recipient, the computation time in Monero to find out the real UTXO is linear with M, while in FAPC, the computation time for receiving is linear with l. So the efficiency for receiving UTXO in FAPC is better than Monero.

Correctness and Security

Theorem 22 (Correctness of TSLA) *For an honest user Alice who wants to pay Bob with her* UTXO_{in}*, she can run TSLA successfully to generate* PK_{out} *as the public key of* UTXO_{out}*, then Bob can recover* SK_{out} *correctly, the auditor can trace Bob's address correctly.*

The security requirements of TSLA are anonymity and traceability, where anonymity can be easily obtained by DDH assumption and anonymity of ring signature. Traceability can be obtained by the hardness assumption of discrete logarithm and the unforgeability of ring signature.

Theorem 23 (Anonymity of TSLA) *For any PPT adversary* $\mathcal{A}$ *without possession of the trapdoor, the advantage of* $\mathcal{A}$ *to correctly guess* $\kappa \in \{1, \cdots, l\}$ *to find out the real recipient is negligible.*

Theorem 24 (Traceability of TSLA) *TSLA has traceability for any PPT adversary* $\mathcal{A}$ *(including the malicious auditor), assuming the unforgeability of ring signature.*

The security models and the proofs are given in the Appendix B.2.

7 Implementation and Performance

We implement FAPC, including the algorithms: TLRS, TRP and TSLA. We also implement Monero as a comparison with FAPC. The implementation is finished in Golang, with Ed25519 curve and Ristretto library. We use SHA256 as the hash function. All experiments are conducted on a desktop with 64-bit Win 7 system and 32GB RAM. The processor is Intel(R) Core(TM) i7-6850K CPU @ 3.6 GHz with 6 cores. We use multi-threading parallel acceleration to improve the efficiency of all schemes.

In our implementation of FAPC, we use AOS' based position-preserving multi-ring signature (PMRS') in TLRS and TSLA, which has better efficiency and compactness than AOS based PMRS. The detailed descriptions of PMRS and PMRS' are in the Appendix A.3. Moreover, we can choose other ring signature as component, such as Ring-CT 3.0, which will bring a significant decrease of size when the ring size is large (e.g. $m, l > 100$), compared to AOS or AOS'. In TRP, we use Borromean ring signature as the component. In the implementation of Monero, we use MLSAG[26] as the linkable ring signature, use Borromean range proof as the range proof.

We give the performance of FAPC and the comparison between FAPC and Monero in 7.1 and give the detailed performance of sub-algorithms in 7.2.

7.1 Performance of FAPC

We compare the transaction size and efficiency between FAPC and Monero for $(1, 1)$ transaction and $(1, 2)$ transaction, where $(\cdot, \cdot)$ denotes the number of input

and output UTXO. In the implementation of FAPC, we choose $m = l = 20$ as the ring size, we choose $n = 32$ as the length of amount. In the implementation of Monero, we choose the same parameter $m = 20, n = 32$. From the performance result we can conclude that the efficiency of FAPC is not far from Monero, and the size of FAPC is almost twice the size of Monero.

Table 1. Performance of FAPC and Monero

Scheme	(in,out)	Size (Byte)	Spend	Verify	Receive	Audit
FAPC	(1,1)	11397	11.24ms	8.03ms	0.29ms	2.39ms
	(1,2)	23206	21.80ms	15.20ms	0.29ms	4.77ms
Monero	(1,1)	5860	6.85ms	6.99ms	26.91ms	N/A
	(1,2)	10180	8.64ms	8.84ms	26.91ms	N/A

7.2 Performance of Sub-algorithms

In this subsection we give the performance of the sub-algorithms of FAPC, including TLRS, TRP and TSLA. The parameter is $m = l = 20, n = 32$, the TLRS is the single ring version.

Table 2. Performance of TLRS, TRP and TSLA

Scheme	Spend	Sign	Prove	Verify	Receive	Trace
TLRS	N/A	0.66ms	N/A	0.57ms	N/A	0.07ms
TRP	N/A	N/A	3.72ms	3.65ms	N/A	2.12ms
TSLA	3.33ms	N/A	N/A	2.45 ms	0.21 ms	0.08ms

8 Conclusion

Lack of audit is the main obstacle to the application of traditional privacy-preserving cryptocurrencies. In this paper, we give the first construction of fully auditable privacy-preserving cryptocurrency against malicious auditors to achieve full auditability and privacy protection. Our construction consists of a traceable and linkable ring signature, a traceable range proof and a tracing scheme for long-term address. The auditor in FAPC can trace the identities of UTXOs (money flows), long-term addresses of users and the amounts in all transactions. FAPC is secure for any PPT adversary with possession of the audit trapdoor, which is an enhanced security requirement and a closer approach towards decentralization. Moreover, the efficiency and compactness of FAPC are very competitive to become a potential option of blockchain-based cryptocurrency in the future.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 415–432. Springer (2002)
2. Au, M.H., Chow, S.S., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: European Public Key Infrastructure Workshop. pp. 101–115. Springer (2006)
3. Back, A.: Ring signature efficiency. Bitcointalk (accessed 1 May 2015) (2015), https://bitcointalk.org/index.php
4. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for r1cs. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 103–128. Springer (2019)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Theory of Cryptography Conference. pp. 60–79. Springer (2006)
6. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. IACR Cryptology ePrint Archive **2019**, 191 (2019)
7. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018)
8. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from dark compilers (2019)
9. Buterin, V.: A next-generation smart contract and decentralized application platform (2014), https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf
10. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: International Colloquium on Automata, Languages, and Programming. pp. 423–434. Springer (2007)
11. Community, D.: Dero white paper. (2019), https://dero.io/attachment/Whitepaper.pdf
12. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 609–626. Springer (2004)
13. Duffield, E., Diaz, D.: Dash: A privacycentric cryptocurrency. GitHub (2015), https://github.com/dashpay/dash/wiki/Whitepaper
14. Facebook: Libra white paper. (2019), https://libra.org/en-US/white-paper/
15. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. IACR Cryptology ePrint Archive **2018**, 990 (2018)
16. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
17. Goodell, B., Noether, S., RandomRun: Compact linkable ring signatures and applications. Cryptology ePrint Archive, Report 2019/654 (2019), https://eprint.iacr.org/2019/654
18. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 253–280. Springer (2015)
19. Jedusor, T.E.: Mimblewimble (2016), http://mimblewimble.cash/20160719-OriginalWhitePaper.txt

20. Li, Y., Yang, G., Susilo, W., Yu, Y., Au, M.H., Liu, D.: Traceable monero: Anonymous cryptocurrency with enhanced accountability. IEEE Transactions on Dependable and Secure Computing (2019)
21. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Transactions on Knowledge and Data Engineering **26**(1), 157–165 (2013)
22. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
23. Maxwell, G.: Confidential transactions (2015), `https://people.xiph.org/~greg/confidential_values.txt`
24. Maxwell, G., Poelstra, A.: Borromean ring signatures (2015), `https://raw.githubusercontent.com/Blockstream/borromean_paper/master/borromean_draft_0.01_34241bb.pdf`
25. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008), `https://git.dhimmel.com/bitcoin-whitepaper/`
26. Noether, S., Mackenzie, A., et al.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
27. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual International Cryptology Conference. pp. 129–140. Springer (1991)
28. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 552–565. Springer (2001)
29. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
30. Sun, S.F., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security. pp. 456–474. Springer (2017)
31. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: International Conference on Information Security Practice and Experience. pp. 48–60. Springer (2005)
32. Van Saberhagen, N.: Cryptonote v 2.0 (2013), `https://cryptonote.org/whitepaper.pdf`
33. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zksnarks without trusted setup. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 926–943. IEEE (2018)
34. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. IACR Cryptology ePrint Archive **2019**, 317 (2019)
35. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. The Computer Journal **56**(4), 407–421 (2013)
36. Yuen, T.H., Sun, S.f., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security (2019)

A Remaining Preliminaries

A.1 AOS Ring Signature

We give the introduction of AOS ring signature[1] in the following:

- $\mathsf{Par} \leftarrow \mathsf{AOS.Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator g as the public parameters.
- $(\mathsf{PK}_\kappa, \mathsf{SK}_\kappa) \leftarrow \mathsf{AOS.KeyGen}(\mathsf{Par})$: according to the public parameters, user P_κ samples $x \in \mathbb{Z}_q^*$ uniformly at random, computes g^x and sets $(\mathsf{PK}_\kappa, \mathsf{SK}_\kappa) = (g^x, x)$.
- $\sigma \leftarrow \mathsf{AOS.Rsign}(\mathsf{SK}_\kappa, \mu, L_{\mathsf{PK}})$: when user P_κ generates a ring signature for message μ, he chooses another $n-1$ users' public keys, together with his own PK_κ to obtain a set of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\}$, where $\mathsf{PK}_\kappa \in L_{\mathsf{PK}}$ and $\kappa \in \{1, \cdots, n\}$, then he does as follows:
  1. P_κ samples $r_\kappa \in \mathbb{Z}_q^*$ uniformly at random, then computes $c_{\kappa+1} = H(g^{r_\kappa}, L_{\mathsf{PK}}, \mu)$;
  2. For $i = \kappa+1, \cdots, n, 1, \cdots, \kappa-1$, P_κ samples $z_i \in \mathbb{Z}_q^*$ uniformly and computes $c_{i+1} = H(g^{z_i}/(\mathsf{PK}_i)^{c_i}, L_{\mathsf{PK}}, \mu)$;
  3. P_κ computes $z_\kappa = r_\kappa + xc_\kappa$;
  4. Output the ring signature $\sigma = (c_1; z_1, \cdots, z_n)$.
- $1/0 \leftarrow \mathsf{AOS.Verify}(\mu, \sigma, L_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, \sigma)$, for $i = 1, \cdots, n$ the verifier computes

$$c_{i+1}^* = H(g^{z_i}/(\mathsf{PK}_i)^{c_i^*}, L_{\mathsf{PK}}, \mu)$$

  where $c_1 = c_1^*$, then checks $c_1 \stackrel{?}{=} c_{n+1}^*$, if all passed then outputs 1, otherwise outputs 0.

A.2 AOS' Ring Signature

AOS' is introduced in the Appendix of [1] with better efficiency than AOS.

- $\mathsf{Par} \leftarrow \mathsf{AOS'.Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator g as the public parameters.
- $(\mathsf{PK}_\kappa, \mathsf{SK}_\kappa) \leftarrow \mathsf{AOS'.KeyGen}(\mathsf{Par})$: according to the public parameters, user P_κ samples $x \in \mathbb{Z}_q^*$ uniformly at random, computes g^x and sets $(\mathsf{PK}_\kappa, \mathsf{SK}_\kappa) = (g^x, x)$.
- $\sigma \leftarrow \mathsf{AOS'.Rsign}(\mathsf{SK}_\kappa, \mu, L_{\mathsf{PK}})$: when user P_κ generates a ring signature for message μ, he chooses another $n-1$ users' public keys, together with his own PK_κ to obtain a set of public keys $L_{\mathsf{PK}} = \{\mathsf{PK}_1, \cdots, \mathsf{PK}_n\}$, where $\mathsf{PK}_\kappa \in L_{\mathsf{PK}}$ and $\kappa \in \{1, \cdots, n\}$, then he does as follows:
  1. For $i = 1, \cdots, \kappa-1, \kappa+1, \cdots, n$, P_κ samples $\alpha, c_i \in \mathbb{Z}_q^*$ uniformly at random, then computes $R = g^\alpha \prod_{i \neq \kappa} \mathsf{PK}_i^{c_i}$ and $c = H(R, L_{\mathsf{PK}}, \mu)$;
  2. P_κ computes $c_\kappa = c - \sum_{i \neq \kappa} c_i$;
  3. P_κ computes $z = \alpha - xc_\kappa$;
  4. Output the ring signature $\sigma = (z; c_1, \cdots, c_n)$.

– $1/0 \leftarrow \mathsf{AOS'.Verify}(\mu, \sigma, L_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, \sigma)$, the verifier computes $R^* = g^z \prod_{i=1}^n \mathsf{PK}_i^{c_i}$, then checks $\sum_{i=1}^n c_i \stackrel{?}{=} H(R^*, L_{\mathsf{PK}}, \mu)$, if all passed then outputs 1, otherwise outputs 0.

Both AOS and AOS' ring signature schemes are unforgeable and anonymous in the random oracle model.

A.3 Position-preserving Multi-ring Signature

AOS and AOS' ring signature scheme can be generalized to be position-preserving multi-ring signatures (PMRS), we give the construction of AOS-based PMRS and AOS'-based PMRS in the following:

AOS-based PMRS

– $\mathsf{Par} \leftarrow \mathsf{PMRS.Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator g as the public parameters.
– $(\mathbf{PK}_\kappa, \mathbf{SK}_\kappa) \leftarrow \mathsf{PMRS.KeyGen}(\mathsf{Par})$: according to the public parameters, user P_κ samples $x_1, \cdots, x_n \in \mathbb{Z}_q^*$ uniformly at random, computes $\mathsf{PK}_{i,\kappa} = g^{x_i}$ and sets $(\mathbf{PK}_\kappa, \mathbf{SK}_\kappa) = ((g^{x_1}, \cdots, g^{x_n}), (x_1, \cdots, x_n))$, where $\mathbf{PK}_\kappa = (\mathsf{PK}_{i,\kappa})_{i\in[1,n]}$ and $\mathbf{SK}_\kappa = (\mathsf{SK}_{i,\kappa})_{i\in[1,n]}$.
– $\sigma \leftarrow \mathsf{PMRS.Rsign}(\mathbf{SK}_\kappa, \mu, \{L_{\mathsf{PK}}^i\}_{i\in[1,n]})$: when user P_κ generates a multi-ring signature for message μ, he chooses another $m-1$ users' public keys, together with his own $\mathbf{PK}_\kappa$ to obtain n rings of public keys $L_{\mathsf{PK}}^i = \{\mathsf{PK}_{i,1}, \cdots, \mathsf{PK}_{i,m}\}$ for $i = 1, \cdots, n$, where $\mathsf{PK}_{i,\kappa} \in L_{\mathsf{PK}}^i$ and $\kappa \in \{1, \cdots, n\}$, then he does as follows:
  1. P_κ samples $r_{i,\kappa} \in \mathbb{Z}_q^*$ uniformly at random for $i = 1, \cdots, n$, then computes $c_{\kappa+1} = H(g^{r_{1,\kappa}}, \cdots, g^{r_{n,\kappa}}, L_{\mathsf{PK}}, \mu)$;
  2. For $j = \kappa+1, \cdots, m, 1, \cdots, \kappa-1$, P_κ samples $z_{i,j} \in \mathbb{Z}_q^*$ uniformly for $i = 1, \cdots, n$ and computes

$$c_{i+1} = H(g^{z_{1,j}}/(\mathsf{PK}_{1,j})^{c_j}, \cdots, g^{z_{n,j}}/(\mathsf{PK}_{n,j})^{c_j}, L_{\mathsf{PK}}, \mu);$$

  3. P_κ computes $z_{i,\kappa} = r_{i,\kappa} + x_i c_\kappa$ for $i = 1, \cdots, n$;
  4. Output the ring signature $\sigma = (c_1; \mathbf{z}_1, \cdots, \mathbf{z}_m)$, where $\mathbf{z}_j = (z_{i,j})_{i\in[1,n]}$.
– $1/0 \leftarrow \mathsf{PMRS.Verify}(\mu, \sigma, L_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, \sigma)$, for $i = 1, \cdots, m$ the verifier computes

$$c_{j+1}^* = H(g^{z_{1,j}}/(\mathsf{PK}_{1,j})^{c_j^*}, \cdots, g^{z_{n,j}}/(\mathsf{PK}_{n,j})^{c_j^*}, L_{\mathsf{PK}}, \mu)$$

  where $c_1 = c_1^*$, then checks $c_1 \stackrel{?}{=} c_{m+1}^*$, if all passed then outputs 1, otherwise outputs 0.

AOS'-based PMRS

- $\mathsf{Par} \leftarrow \mathsf{PMRS'.Setup}(\lambda)$: system chooses an elliptic curve $\mathbb{G}$ and a generator g as the public parameters.
- $(\mathbf{PK}_\kappa, \mathbf{SK}_\kappa) \leftarrow \mathsf{PMRS'.KeyGen}(\mathsf{Par})$: according to the public parameters, user P_κ samples $x_1, \cdots, x_n \in \mathbb{Z}_q^*$ uniformly at random, computes $\mathsf{PK}_{i,\kappa} = g^{x_i}$ and sets $(\mathbf{PK}_\kappa, \mathbf{SK}_\kappa) = ((g^{x_1}, \cdots, g^{x_n}), (x_1, \cdots, x_n))$, where $\mathbf{PK}_\kappa = (\mathsf{PK}_{i,\kappa})_{i\in[1,n]}$ and $\mathbf{SK}_\kappa = (\mathsf{SK}_{i,\kappa})_{i\in[1,n]}$.
- $\sigma \leftarrow \mathsf{PMRS'.Rsign}(\mathbf{SK}_\kappa, \mu, L_{\mathsf{PK}})$: when user P_κ generates a multi-ring signature for message μ, he chooses another $m-1$ users' public keys, together with his own $\mathbf{PK}_\kappa$ to obtain n rings of public keys $L^i_{\mathsf{PK}} = \{\mathsf{PK}_{i,1}, \cdots, \mathsf{PK}_{i,m}\}$ for $i = 1, \cdots, n$, where $\mathsf{PK}_{i,\kappa} \in L^i_{\mathsf{PK}}$ and $\kappa \in \{1, \cdots, m\}$, then he does as follows:
  1. For $j = 1, \cdots, \kappa-1, \kappa+1, \cdots, m$, P_κ samples $c_j \in \mathbb{Z}_q^*$ uniformly at random, then for $i = 1, \cdots, n$, P_κ samples $\alpha_i \in \mathbb{Z}_q^*$ uniformly at random, then he computes $R_i = g_i^\alpha \prod_{j\neq\kappa} \mathsf{PK}_{i,j}^{c_j}$ for $i = 1, \cdots, n$, then computes $c = H(R_1, \cdots, R_n, L_{\mathsf{PK}}, \mu)$;
  2. P_κ computes $c_\kappa = c - \sum_{j\neq\kappa} c_j$;
  3. P_κ computes $z_i = \alpha_i - x_i c_\kappa$ for $i = 1, \cdots, n$;
  4. Output the ring signature $\sigma = (z_1, \cdots, z_n; c_1, \cdots, c_m)$.
- $1/0 \leftarrow \mathsf{PMRS'.Verify}(\mu, \sigma, L_{\mathsf{PK}})$: for a ring signature $(\mu, L_{\mathsf{PK}}, \sigma)$, the verifier computes $R_i^* = g^{z_i} \prod_{j=1}^m \mathsf{PK}_{i,j}^{c_j}$, then checks $\sum_{j=1}^m c_j \stackrel{?}{=} H(R_1^*, \cdots, R_n^*, L_{\mathsf{PK}}, \mu)$, if all passed then outputs 1, otherwise outputs 0.

A.4 Borromean Range Proof

Borromean range proof is used in Monero to provide the validity proof of transaction amount ($a \in [0, 2^n - 1]$) by making use of Borromean ring signature and Pedersen commitment:

- $\mathsf{BRP.Setup}$: System chooses public parameters $(\mathbb{G}, q, g, h)$.
- $\mathsf{BRP.Prove}$:
  1. According to the public parameters, amount $a \in [0, 2^n - 1]$, prover computes the commitment $c = g^x h^a$;
  2. Prover computes the binary expansion $a = a_0 + \cdots + 2^{n-1} a_{n-1}, a_i = 0, 1$ for $i = 0, \cdots, n-1$;
  3. Prover samples samples $x_0, \cdots, x_{n-1}$ uniformly, computes $\beta = x - x_0 - \cdots - x_{n-1}$;
  4. For every $i = 0, \cdots, n-1$, prover computes $c_i = g^{x_i} h^{2^i a_i}, c_i' = g^{x_i} h^{2^i a_i - 2^i}$, prover outputs $L^i_{\mathsf{PK}} = (c_i, c_i')$;
  5. Prover generates n sets of PKs by $L = \{L^0_{\mathsf{PK}}, \cdots, L^{n-1}_{\mathsf{PK}}\}$;
  6. Prover runs Borromean ring signature $\sigma \leftarrow \mathsf{Rsign}(L, \mathsf{SK}, c)$, where $\mathsf{SK} = (x_0, \cdots, x_{n-1})$, outputs $\pi_{\mathsf{BP}}(c) = (L, \sigma, c, \beta)$.
- Verify: For $i = 0, \cdots, n-1$, verifier checks as follows:
  1. Checks $g^\beta \cdot \prod c_i \stackrel{?}{=} c$ and $c_i / c_i' \stackrel{?}{=} h^{2^i}$;
  2. Checks the validity of Borromean ring signature σ, if all passed then outputs 1, otherwise outputs 0.

A.5 Key generation algorithm for UTXO in Monero

In the Monero system, user's (Bob's) long-term address $\mathsf{Add} = (A, B)$ consists a view key $A = g^a$ and a spending key $B = g^b$, where $\mathsf{ASK} = (a, b)$ is the corresponding secret key of the address. When Alice pay Bob with her money, Alice generates $r \leftarrow \mathbb{Z}_q^*$, computes and outputs $(R = g^r, \mathsf{PK}_{out} = g^{H(A^r)} \cdot B)$, where PK_{out} is the new UTXO's public key. When receiving money, Bob checks $\mathsf{PK}_{out} \stackrel{?}{=} g^{H(R^a)} \cdot B$, if passed then computes $\mathsf{SK}_{out} = H(R^a) + b$ and receives the new UTXO to his wallet. The correctness is obtained from $R^a = g^{ar} = A^r$, which is the Diffie-Hellman key exchange.

B Remaining Proofs

B.1 Traceability Proof of TRP

Proof (Traceability of TRP). For any PPT adversary $\mathcal{A}$ without possession of the trapdoors, when $\mathcal{A}$ finished the tracing game with $\mathcal{S}$ in Definition 11, $\mathcal{A}$ generates a commitment c for a hidden amount a and range proof $\pi_{\mathsf{TRP}}(c) = (c, \beta, \{c_i, c_i', \mathsf{TK}_i, I_i\}_{i\in[0,n-1]}, \sigma)$, We assume that $\mathcal{A}$ wins the tracing game with nonnegligible advantage δ, that is, $\pi_{\mathsf{TRP}}(c)$ satisfying the following:

$$\mathsf{TRP.Verify}(\pi_{\mathsf{TRP}}(c)) = 1 \text{ and } \mathsf{TRP.Trace}(\pi_{\mathsf{TRP}}(c), y) \neq a.$$

According to the soundness of TRP, we know $c = g^x h^a$ with $a \in [0, 2^n - 1]$ and $c_i = g^{x_i} h^{2^i a_i}$ with $a_i = 0, 1$ for every $i = 0, \cdots, n-1$, satisfying $\sum 2^i a_i = a$ and $\beta + \sum x_i = x$, except for negligible probability $\epsilon_0 = negl(\lambda)$, we can set $\mathsf{TK}_i = g^{s_1} h^{t_1}$ and $I_i = g^{s_2} h^{t_2}$, then we get $e_k = H(\{c_i, \mathsf{TK}_i, I_i\}_{i\in[0,n-1]}, k)$ for $k = 1, 2$ and $L^i_{\mathsf{PK}} = \{c_i \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}, c_i' \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2}\} = \{\mathsf{PK}_i, \mathsf{PK}_i'\}$. Without loss of generality we assume PK_i is the corresponding signing key, then

$$c_i \cdot \mathsf{TK}_i^{e_1} \cdot I_i^{e_2} = g^{x_i + s_1 e_1 + s_2 e_2} h^{2^i a_i + t_1 e_1 + t_2 e_2} = (g^{1+ye_1} h^{e_2})^z = \mathsf{PK}_i$$

with z being known to $\mathcal{A}$ except for negligible probability $\epsilon_1 = negl(\lambda)$, under the unforgeability of Borromean multi-ring signature. Similar to Theorem 19, we can get $x_i = z = t_2, s_1 = yz, s_2 = 0, t_1 = 0, a_i = 0$ with nonnegligible probability, then $c_i^y = g^{x_i y} = \mathsf{TK}_i$ implies $a_i^* = a_i$ for $i = 0, \cdots, n-1$, then $\mathsf{TRP.Trace}(\pi_{\mathsf{TRP}}(c), y) = a$, which contradicts with the assumptions before, then we get the traceability of TRP. □

B.2 Proofs of TSLA

Correctness

Proof (Correctness of TSLA).

For Bob's address $\mathsf{Add} = (A, B) = (g^{x_v}, g^{x_s})$. Assume Alice has finished the TSLA and outputs $(\mathsf{PK}_{out}, R, R_1, R_2)$ as the output, and we know that $\mathsf{PK}_{out} = g^z \cdot B, R = A^z, R_1 = h_1^z, R_2 = h_2^z$. Then Bob can find his Add in L_{Add} and

computes $B = \mathsf{PK}_{out}/R^{x_v^{-1}}$ to find out if he is the receiver, then Bob can get $\mathsf{SK}_{out} = z + x_s$, where z is obtained by decryption from ct.

For the auditor, he can compute $B = \mathsf{PK}_{out}/R_1^{y^{-1}}$ to trace the address of Bob successfully. □

Security Model Assuming there is an adversary who is given access to oracle $\mathcal{RO}$, $\mathcal{JO}$ and $\mathcal{CO}$, in which the $\mathcal{JO}$ (join new address) and $\mathcal{CO}$ (get the address secret key) are similar to TLRS, we give the definition of anonymity of TSLA in the following:

Definition 25 (Anonymity of TSLA) *Anonymity of TSLA is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$ and $\mathcal{CO}$. $\mathcal{A}$ gives a set of addresses $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$ in which all the addresses in L_{Add} are returned by $\mathcal{A}$ to $\mathcal{JO}$, and none of them are queried by $\mathcal{A}$ to $\mathcal{CO}$, $\mathcal{S}$ randomly picks $\kappa \in \{1, \cdots, l\}$, computes $(\mathsf{PK}_{out}, R, R_1, R_2, \theta) \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add}_\kappa)$ and sends $(\mathsf{PK}_{out}, R, R_1, R_2, \theta)$ to $\mathcal{A}$, then $\mathcal{A}$ outputs a guess $\kappa^* \in \{1, \cdots, l\}$. $\mathcal{A}$ wins the game if he successfully guesses $\kappa^* = \kappa$.*

We give the advantage of $\mathcal{A}$ in anonymity attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{anon} = |\Pr[\kappa^* = \kappa] - 1/l|.$$

TSLA is anonymous if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{anon} = negl(\lambda)$.

We also give the definition of traceability of TSLA in the following:

Definition 26 (Traceability of TSLA) *Traceability of TSLA is defined in the following game between the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$, simulator $\mathcal{S}$ runs Setup to provide the public parameters for $\mathcal{A}$, $\mathcal{A}$ is given access to oracles $\mathcal{RO}$, $\mathcal{JO}$, $\mathcal{CO}$. $\mathcal{A}$ generates a set of addresses $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$, $\mathcal{A}$ wins the game if he successfully generates a valid TSLA output*

$$(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R, R_1, R_2, \theta) \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add}_\kappa)$$

using $\mathsf{Add}_\kappa \in L_{\mathsf{Add}}$, satisfying the following:

1. $\mathsf{TSLA.Verify}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R, R_1, R_2, \theta) = 1$.
2. $\mathsf{Add}_i \neq \mathsf{Add}_j$ *for* $1 \leq i < j \leq l$.
3. $\mathsf{TSLA.Trace}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R_1, y) \neq \kappa$ *or* $\mathsf{TSLA.Trace}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R_1, y) = \perp$.

We give the advantage of $\mathcal{A}$ in tracing attack as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{trace} = \Pr[\mathcal{A} \text{ wins}].$$

TSLA scheme is traceable if for any PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{trace} = negl(\lambda)$.

Anonymity

Proof (Anonymity of TSLA). For $\mathsf{Add}_i = (A_i, B_i) = (g^{a_i}, g^{b_i})$, $i = 1, \cdots, l$, we know from $(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R, R_1, R_2, \theta) \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add}_\kappa)$ that there exists $z \in \mathbb{Z}_q^*$, satisfying $R = A_\kappa^z, R_1 = h_1^z, R_2 = h_2^z$, $\mathsf{PK}_{out} = g^z \cdot B_\kappa$ and

$$\theta \leftarrow \mathsf{Rsign}(L_{\mathsf{Add}}, L_1, L_2, g_1, g_2, z).$$

From the anonymity of ring signature, we know the advantage of $\mathcal{A}$ to correctly guess κ is negligible. Then we only need to prove $(\mathsf{PK}_{out}, R, R_1, R_2)$ and $\mathbf{r} = (r_1, r_2, r_3, r_4) \in \mathbb{G}^4$ are computational indistinguishable, where $\mathbf{r}$ is sampled from uniform distribution U.

Since $(\mathsf{PK}_{out}, R, R_1, R_2) = (g^z \cdot B_\kappa, A_\kappa^z, h_1^z, h_2^z)$, from the extended DDH assumption we know that $(g^z, A_\kappa^z, h_1^z, h_2^z)$ and $\mathbf{r}$ are computational indistinguishable, then we know $(g^z \cdot B_\kappa, A_\kappa^z, h_1^z, h_2^z)$ and $(r_1 \cdot B_\kappa, r_2, r_3, r_4)$ are also computational indistinguishable. Note that the distribution of $(r_1 \cdot B_\kappa, r_2, r_3, r_4)$ and (r_1, r_2, r_3, r_4) are identical, then we know that $(\mathsf{PK}_{out}, R, R_1, R_2)$ and (r_1, r_2, r_3, r_4) are computational indistinguishable. We know the probability of $\mathcal{A}$ to correctly guess κ is $1/l$ when the input is $(r_1, r_2, r_3, r_4, \theta)$ (θ can be obtained by programming the random oracle), then we finish the anonymity proof of TSLA. □

Traceability

Proof (Traceability of TSLA). For a PPT adversary $\mathcal{A}$ with possession of the trapdoor y, when $\mathcal{A}$ finished the tracing game with $\mathcal{S}$ in Definition 26, $\mathcal{A}$ generates a set of addresses $L_{\mathsf{Add}} = \{\mathsf{Add}_1, \cdots, \mathsf{Add}_l\}$, we assume that $\mathcal{A}$ wins the tracing game with nonnegligible probability δ, that is, $\mathcal{A}$ generates a TSLA output $(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R, R_1, R_2, \theta) \leftarrow \mathsf{TSLA.Spend}(\mathsf{Add}_\kappa)$ using $\mathsf{Add}_\kappa \in L_{\mathsf{Add}}$, satisfying the following:

1. $\mathsf{TSLA.Verify}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R, R_1, R_2, \theta) = 1$.
2. $\mathsf{Add}_i \neq \mathsf{Add}_j$ for $1 \leq i < j \leq l$.
3. $\mathsf{TSLA.Trace}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R_1, y) \neq \kappa$ or $\mathsf{TSLA.Trace}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R_1, y) = \perp$.

Without loss of generality, we assume $\mathsf{PK}_{out} = g^{s_1} h_2^{t_1}$, $R = g^{s_2} h_2^{t_2}, R_1 = g^{s_3} h_2^{t_3}, R_2 = g^{s_4} h_2^{t_4}$ and $\mathsf{Add}_i = (A_i, B_i) = (g^{a_i}, g^{b_i})$, then we get

$$L_1 = \{\mathsf{PK}_{out} \cdot B_1^{-1} \cdot R_1^{e_1} \cdot R_2^{e_2}, \cdots, \mathsf{PK}_{out} \cdot B_l^{-1} \cdot R_1^{e_1} \cdot R_2^{e_2}\}$$

$$= \{g^{s_1 - b_1 + s_3 e_1 + s_4 e_2} h_2^{t_1 + t_3 e_1 + t_4 e_2}, \cdots, g^{s_1 - b_l + s_3 e_1 + s_4 e_2} h_2^{t_1 + t_3 e_1 + t_4 e_2}\}.$$

And $L_2 = \{A_1 \cdot h_1^{e_1} \cdot h_2^{e_2}, \cdots, A_l \cdot h_1^{e_1} \cdot h_2^{e_2}\} = \{g^{a_1 + y e_1} h_2^{e_2}, \cdots, g^{a_l + y e_1} h_2^{e_2}\}$, where $e_i = H(L_{\mathsf{Add}}, R, R_1, R_2, \mathsf{PK}_{out}, i)$ for $i = 1, 2$. We also know that the generator for L_1 is $g_1 = g \cdot h_1^{e_1} \cdot h_2^{e_2}$ and for L_2 is $g_2 = R \cdot R_1^{e_1} \cdot R_2^{e_2}$.

Under the unforgeability of the position-preserving double-ring signature, we know there exists $z_1, z_2 \in \mathbb{Z}_q^*$ known by $\mathcal{A}$, and $\kappa \in \{1, \cdots, l\}$ such that $g^{s_1 - b_\kappa + s_3 e_1 + s_4 e_2} h_2^{t_1 + t_3 e_1 + t_4 e_2} = g_1^{z_1}$ and $A_\kappa \cdot h_1^{e_1} \cdot h_2^{e_2} = g_2^{z_2}$. Using similar arguments as Theorem 14, we can get $z_1 = z_2^{-1} = z, s_1 = z + b_\kappa, s_2 = a_\kappa z, s_3 =$

$yz, s_4 = t_1 = t_2 = t_3 = 0, t_4 = z$ with nonnegligible probability, then $\mathsf{PK}_{out} = g^z B_\kappa$ and $R_1 = g^{yz} = h_1^z$, then we have $B_\kappa = \mathsf{PK}_{out}/R_1^{y^{-1}}$, which means $\mathsf{TSLA.Trace}(L_{\mathsf{Add}}, \mathsf{PK}_{out}, R_1, y) = \kappa$, which contradicts with the assumptions before, then we get the traceability of TSLA. □